Creating systems to manage information 2

Getting to know your unit

Assessment

You will be externally assessed by a task.

Information technology is all about managing and manipulating information. Database software applications are very widely used to support many business processes and internet facilities. Google™, Facebook™ and YouTube™ are all based around a database, as are many other applications.

In this unit, you will explore the purpose and structure of relational databases. You will look at the structure of data. You will look at how you can design and then implement an effective database system solution. Note that, though the screenshots in this unit show a particular database, the information in this unit can be applied to whichever database you are using.

How you will be assessed

This unit is externally assessed by a practical task which will be completed under supervised conditions for 10 hours in a 1 week period which can be arranged over a number of sessions. You will be assessed on your ability to design, create, test and evaluate a relational database system to manage information.

Grade descriptors

To achieve a grade, a learner is expected to demonstrate these attributes across the essential content of the unit. The maximum number of marks for this unit is 66.

To pass the unit

- You will be able to use your knowledge and understanding of database design and development terminology, standards, concepts and processes.
- You can apply problem-solving skills to design and develop a solution in context.
- You can demonstrate your understanding of how to use standard database. constructs to develop a functioning solution that evidences testing and evaluation.

To achieve a distinction

- You can evaluate a given problem and develop a detailed and complex solution to meet all requirements of the brief.
- You can apply an in-depth understanding of database constructs, using test results to produce an optimised solution.
- You are able to evaluate the quality, performance and usability of your database with supporting justification.

Assessment outcomes

AO1 Demonstrate knowledge of database development terminology, standards, concepts and processes

AO2 Apply knowledge and understanding of database development terminology, standards, concepts and processes to create a software product to meet a client brief

AO3 Analyse information about database problems and data from test results to optimise the performance of a database solution

AO4 Evaluate evidence to make informed judgements about the success of a database's design and performance

AO5 Be able to develop a database solution to meet a client brief with appropriate justification

Getting started

Data is all around us, your social media newsfeed, mobile phone call list, the weather forecast, your college or school timetable. Data becomes information when it is useful to us and we can use it in a meaningful way. Think of all the sources of data that you have accessed recently. Were they useful to you? Could they have been presented in a different way to make them more useful? What would be the implications if the data you accessed had been inaccurate? In this unit, you will develop an understanding of database technology which is central to how so much of the internet-centric world around us operates. This understanding will be essential to you as you work towards a career in Information Technology.

A The purpose and structure of relational database management systems

Databases are used to provide a wide range of applications, from e-commerce and customer billing through to games and social media. In this section, you will look at the purpose and structure of relational database management systems.

Relational database management systems

First, we will cover the types of relational database management systems (RDBMS).

Types of RDBMS and their characteristics

There are a number of different types of RDMBS, which can be broadly categorised in the following way:

- Desktop systems – These are designed for personal use. The best known of these is Microsoft® Access®, which we will use throughout this unit. Microsoft® Access® is part of the Microsoft® Office® suite and is focused on single-user or small-departmental applications. Access® includes an end user interface and features such as 'Wizards' to simplify the creation of databases for end users. Access® only runs on computers with the Windows® operating system.
- Client–server database systems – These are designed for distributed multi-user databases with the database files stored on a server computer. Databases in this category are developer tools and they need a lot more developing before they include the end user interface features found in Access®. This category of RDBMS can be further divided into the following.
  - Open source database systems – The best known and most widely used **open source** database system is MySQL™. MySQL™ runs on a wide range of operating systems including Windows®, Linux™ and Apple® OSX, and is a popular choice for web applications.
  - Proprietary database systems – The most widely used **proprietary** database systems are Microsoft®'s SQL Server®, Oracle® Database and IBM®'s DB2®.

Key term

Open source – Open source software has its source code available for the general public to use and modify as they desire, free of charge. This is rather than the source code being kept as a commercial secret by the company that created it. Open source software is meant to be a collaborative effort by programmers who give their time to develop the application.

Proprietary – In the context of software, proprietary means the software is owned by a commercial organisation which develops it and sells a licence to use the software. Microsoft® Office® is an example of proprietary software, whereas Open Office is a free-to-use open source office software suite.

Research

MySQL® and Oracle® Database are the two most widely used database software applications. What are the differences and similarities between them?

Relational data structures

To understand the relational model, it is a good idea to start by thinking of simple real-life data examples. For example, if you drew up a list of Christmas presents to buy your family, it might look something like the list in Figure 2.1.

Figure 2.1: Christmas present list

If you wanted to, you could create a budget for your Christmas spending by putting this data into a spreadsheet and adding up the cost of the items. This is often called a flat file database as it contains a single list of data (in database terminology, this is a table). Please read the worked example, which will explain how two tables of data are used to create a relational database, before going on to learn more about relational database terminology.

In *relational database* terminology, a table of data is known as a *relation*. Every relation has columns which represent the different data that is stored in the relation. In the customer details table in our example, this would be items like customer number, first name, surname, house number and postcode. Each of these items is called an *attribute*. Each attribute has a range of allowable values, which is known as a *domain*. An example of the attributes and domains for the customer details table are shown in Table 2.1.

Table 2.1: Attributes and domains in the customer details table

Attribute	Domain name	Domain definition
Customer number	CUST_NO	Numeric, 6 digits
Customer first name	F_Name	Character, size 25
Customer surname	S_Name	Character, size 25
Address line 1	Addr_1	Character, size 25
Address line 2	Addr_2	Character, size 25
Town	Town	Character, size 25
Postcode	Postcode	Character, size 8

Worked example: Ordering from an online store

Imagine that you ordered all of your Christmas presents (see Figure 2.1) from an online store. Due to the fact that you might place several orders with the same online store, they would keep at least two tables of data. One table would hold the information about you, the customer, and the other table would hold information about the orders that you have placed.

In theory this data could all be held in one table, but then every time you placed an order you would need to include all of your details (such as name and address) again. This would mean duplicating a lot of data; therefore, it is a much better solution to have two tables.

However, there needs to be a way to link (relate) the two tables (those holding your customer details and other holding your orders) together because the online store needs to know which customer has placed each order. We could use the customer's name to relate the two tables but, as there might be several people with the same name, a better idea is to use a unique identifier for each customer, such as a customer number. Therefore, every order that a particular customer places is identified by including the customer number, from the *Customer* table, in the orders placed by that customer in the *Orders* table. The two tables are said to be related by the customer number.

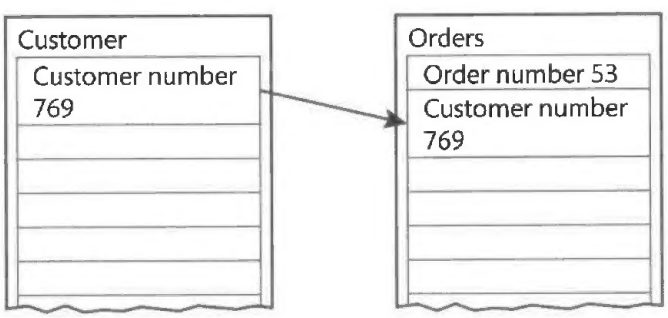


Figure 2.2: The Customer and Orders tables

The number of attributes that a relation contains is known as its *degree*. The degree of the customer details relation is 7 (because there are seven attributes listed in the table). The domain definition describes the type and size of the data to be stored in that attribute; this is often called the *datatype*.

Key term

Datatype – Datatypes are used in both programming and in database development as the definition of the type of data that is to be stored (in the case of database) in a field. The most commonly used datatypes are text and number, but there are variations on these and other datatypes too.

Research

What datatypes does Microsoft Access® support? What kind of data can they store? How might you use the different datatypes?

The rows within a relation represent complete sets of attributes, sometimes called records but, in relational database terminology, they are called *tuples*. The uniqueness of the attributes within a relation is called the *cardinality* of the relation. In Table 2.1, the Town attribute is likely to have the same value in many records (e.g. London or Manchester) as there are a limited number of towns within the UK, whereas the Cust_no attribute is different for each record so its cardinality is equal to the number of records in the customer table. Access® and other database software applications sometimes use different terminology to the standard relational database terminology. Table 2.2 compares the standard relational database terminology with that used by Access®.

Table 2.2: Comparison between standard relational database terminology and that used by Access®

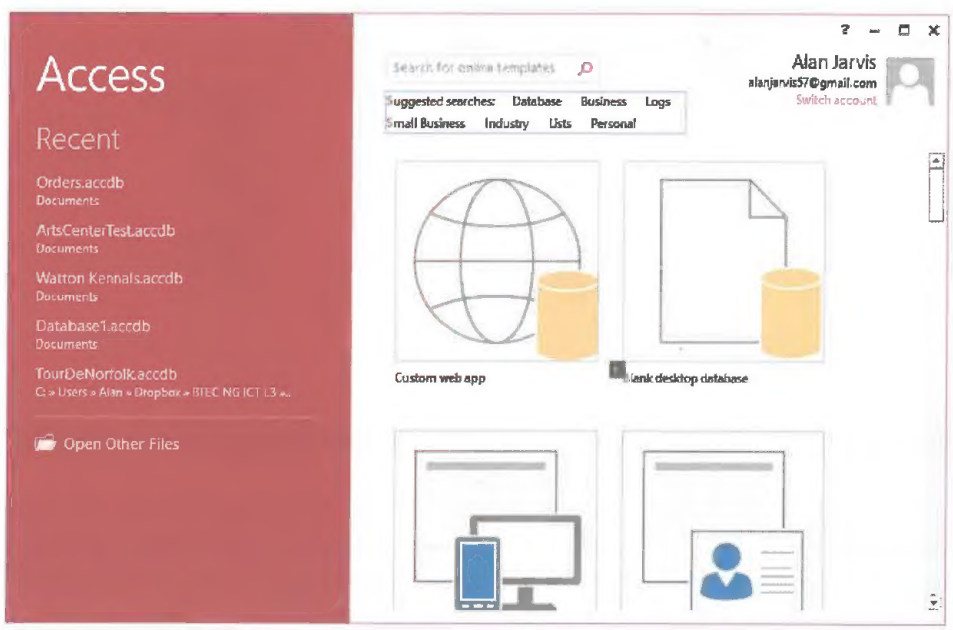
Standard relational database terminology	Access® terminology for commonly used terms that differs from the standard
Relation	Table
Attribute	Field
Tuple	Record

In this book, to avoid confusion, we will use the terminology used in Access®, because it is likely that you will use this database software application to complete this unit.

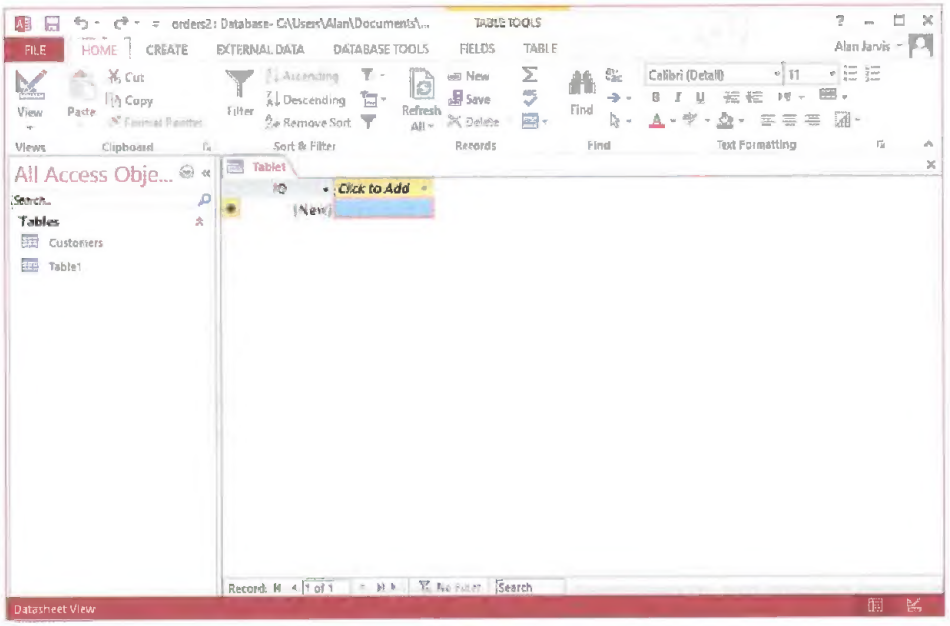
Step by step: Creating tables

8 Steps

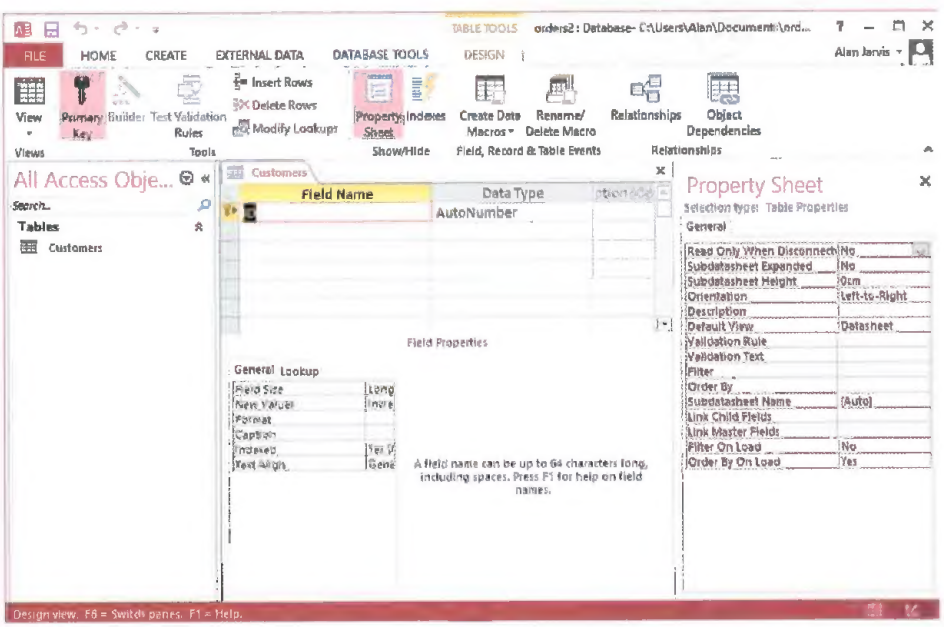
1 When using Access®, all the tables within a single database are stored in the same database file. When you start Access®, it displays the screen shown. From this screen, to create a new database file to which tables can be added, click the Blank desktop database icon.



2 You will then be asked for a filename and a location for the database file. Once you have done this, you will see this screen.

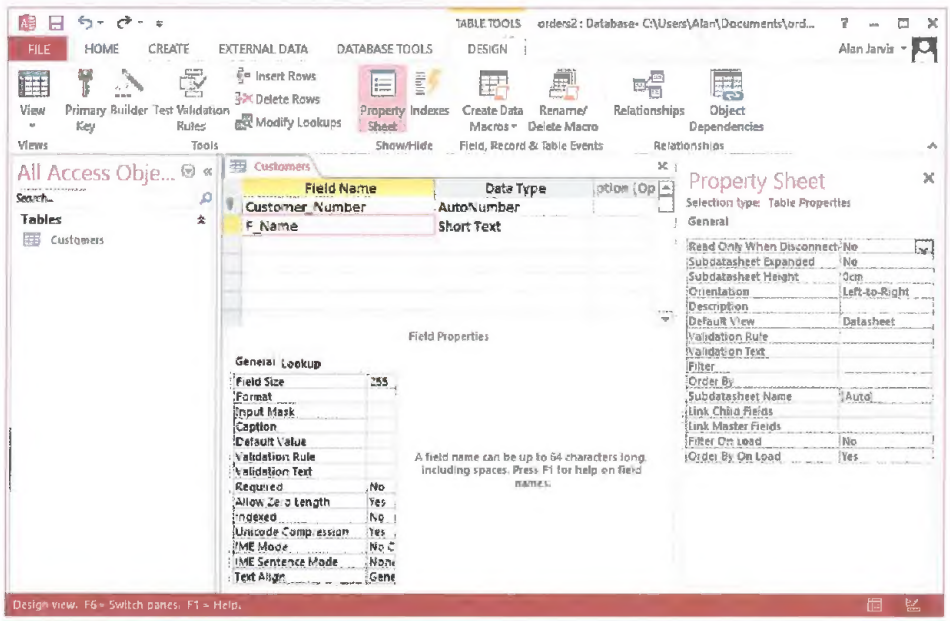


3 Access® creates a table for you and you can complete the fields in this Datasheet View. To get more control over how the table is structured, it is best to swap to Design View, so click the View icon on the far left of the toolbar and choose Design View from the drop-down menu. Access® will now ask you to save the table. Type a meaningful table name (e.g. Customers) and click OK. You will now see this display.

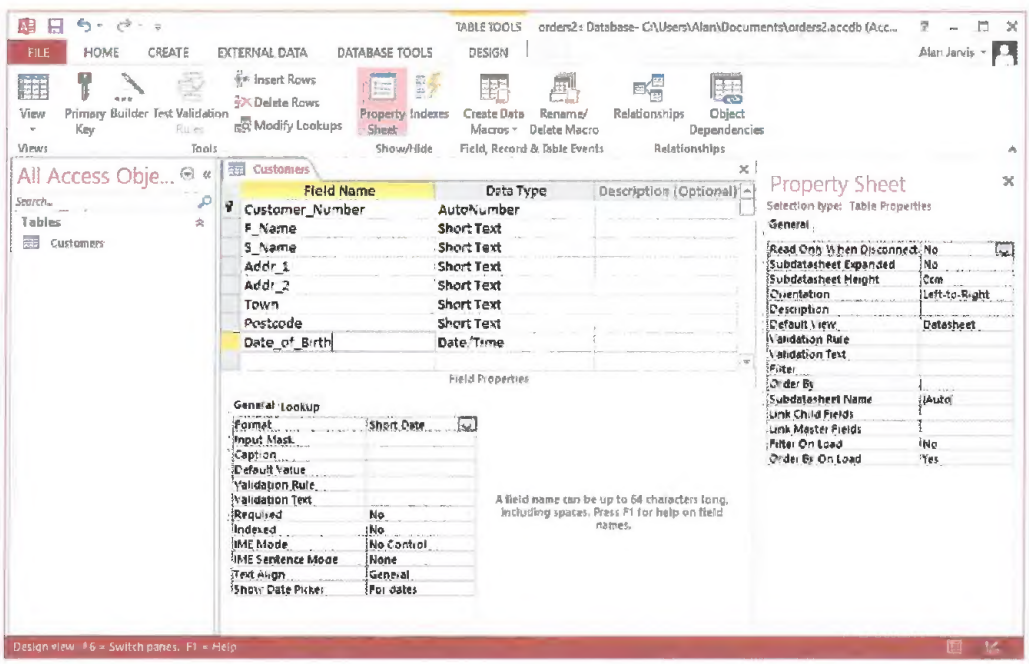


4 Access® creates a field, that it calls 'ID' for you, which it sets as the primary key. It has a datatype of AutoNumber, which means that Access® will automatically give each record in the table a unique consecutive number, starting from 1. This can be a good idea for a primary key field, as we will see later, but 'ID' is not a good name. Something more meaningful like 'Customer_Number' would be a better choice.

5 The next field to be added is 'F_Name' in the second row of the field name list. Access® automatically adds the datatype of 'Short Text', which allows up to 255 characters (which should be more than enough characters). The display should now look like this.



6 Under the field name grid you can see the properties for whichever field you have selected in the grid. Here, for example, you could adjust the field size, if required, to the 255 characters listed in the example in step 5. All the other fields from that example are text so they can be added in the same way as the F_Name field. However, suppose that you also wanted to add the customer's date of birth. This time, where Access® adds the default datatype of Short Text, you need to click in the box and choose Date/Time from the drop-down menu, as shown here.



7 In the properties for this field, you can see that there is no size property, but there is a format property. Clicking in the Date/Time box will drop down a list of different date and time formats. In this case, the 'Short date' format (e.g. 20/12/2015) is fine. You can add as many fields as you like. Once they are all added, you can swap to Datasheet View and try entering some data. Click on the View icon in the toolbar and choose Datasheet View (Access® will require you to save the table design first).

8 Once in Datasheet View, you cannot type in the Customer_Number field as it is an AutoNumber field, meaning that Access® completes it for you. You can type in all the other fields but, because the 'Date_of_Birth' field is formatted as a date, you must enter a valid date, otherwise you will get an error message.

Tip

It is very important that you check that your table design is correct, particularly with regard to datatypes, before you add large numbers of records. Correcting design mistakes later, when the table is full of data, can be complex and time consuming.

Relational algebra

The theoretical basis behind relational databases and query languages such as **SQL** is known as relational algebra. The practical applications of relational algebra are database queries (or relational algebra operations) that allow us to extract useful information from the tables within a database.

Key term

SQL – stands for structured query language and is a special purpose programming language used for managing data in relational databases. SQL is a standard language (developed by ISO, the International Organisation for Standardisation) used by all the main database programs.

There are a number of different relational algebra operations that we can apply to tables. Some simple examples are given below. Each operation is represented by a special *symbol*.

Select

The *select* operation involves selecting a subset of records from a table that match certain criteria. It is written as

$$\sigma \text{ condition}^{(\text{relation})}$$

The 'condition' is a Boolean expression in which rows are selected if true. For example, suppose you had a table (relation) which contains books, and one of the fields is the subject, then the following expression will display all the books with a subject of "computing"

$$\sigma \text{ Subject} = \text{"computing"}^{(\text{books})}$$

Join

The *join* operation, as the name suggests, involves joining two tables together. There are several versions of the join operation, the simplest of which is the *natural join*. The symbol for natural join is ⋈. To carry out a natural join, the two tables must have at least one common field. For example, imagine a college database with a table of all the courses, as shown in Table 2.3 (just a few fields are shown for simplicity).

Table 2.3: Table of all the Courses offered by a college

Course ID	Course name	Department
001	BTEC IT L3	Computing
002	HNC Child care	Health and social care
003	A level Business	Business studies
004	A Level Computing	Computing

The second table holds details of the college departments and who is the head of each department, as shown in Table 2.4 (again just a few fields are shown for simplicity).

Table 2.4: Table of all the Departments at a college and the names of the heads of departments

Department	Head of department
Computing	Wendy Smith
Health and social care	Alan Jones
Business studies	Mohammed Ahmed

Therefore the join operation *Courses* ⋈ *Departments* would produce the following, see Table 2.5.

Table 2.5: Table produced by join operation Courses ⋈ Departments

Course ID	Course name	Department	Head of department
001	BTEC IT L3	Computing	Wendy Smith
002	HNC Child care	Health and social care	Alan Jones
003	A level Business	Business studies	Mohammed Ahmed
004	A Level Computing	Computing	Wendy Smith

Union

The *union* operation requires the two tables to have the same fields, and it involves taking all the records that are in one table or in the other table (or in both) and putting them together. Duplicate records are eliminated. The symbol for union is ∪. For example, imagine two separate tables containing customer details (see Tables 2.7 and 2.8).

Table 2.6: Customer_table1

Name	Address	Postcode	Telephone number
J Smith	10 High Street	NR6 2ZZ	01883 12345
A Jones	22 Station Road	IP95 1QQ	01999 22222
B Williams	105 London Road	N65 6XX	01888 33333
G Anderson	4 North Lane	E88 4VV	02946 11133

Table 2.7: Customer_table2

Name	Address	Postcode	Telephone number
A Jones	22 Station Road	IP95 1QQ	01999 22222
S Evans	165 Main Avenue	AB69 4KK	0345 998877
G Anderson	4 North Lane	E88 4VV	02946 11133
T Thomas	52 Harbour Lane	SW8 9ZQ	01998 88888

Therefore the *union* operation *Customer_table1* ∪ *Customer_table2* would produce the following (see Table 2.8).

Table 2.8: Table produced by union operation Customer_table1 ∪ Customer_table2

Name	Address	Postcode	Telephone number
J Smith	10 High Street	NR6 2ZZ	01883 12345
A Jones	22 Station Road	IP95 1QQ	01999 22222
B Williams	105 London Road	N65 6XX	01888 33333
G Anderson	4 North Lane	E88 4VV	02946 11133
S Evans	165 Main Avenue	AB69 4KK	0345 998877
T Thomas	52 Harbour Lane	SW8 9ZQ	01998 88888

Intersect

The *intersect* operation is similar to the union operation but only those records that appear in both the tables are selected (the records where the two tables intersect or overlap). The symbol for intersect is ?. For example, using the same two customer details tables from the union operation section, the intersect operation *Customer_table1* **?** *Customer_table2* would produce the following (see Table 2.9).

Table 2.9: Table produced by intersect operation Customer_table1 ? Customer_table2

Name	Address	Postcode	Telephone number
A Jones	22 Station Road	IP95 1QQ	01999 22222
G Anderson	4 North Lane	E88 4VV	02946 11133

Database relations

We have already mentioned how, in a relational database, tables can be related to each other. These relationships between tables model real-world relationships and are called entity relationships.

Entity relationships

Entities (table) are real-world things like customers, books, courses, authors and teachers. Entities often have relationships with each other which are called entity relationships. So for example, customers order goods, books are written by authors and courses are taught by teachers.

There are three types of entity relationships, the most common of which is one-to-many.

One-to-many entity relationships

Look back at the Worked example: Ordering from an online store. The example we used was of a customer who ordered goods from an online store and the store recorded the order, but kept the data about the customer in a separate table to the data about the order.

The tables in that example were linked using the customer number. This means that the customer details only need to be recorded once, no matter how many orders the customer places. Therefore one customer record can be related to as many different order records as the customer places. However, in the Orders table, each record is related to one and only one customer, since the same order cannot be placed by two different customers. This is called a *one-to-many relationship* (between one customer and many orders). To help conceptualise databases designs, it is common to draw a diagrammatic representation of the relations and the relationships between them using an *entity-relationship diagram (ERD)*. The relationship between the Customer and the Orders tables (Figure 2.2) would be drawn as follows (see Figure 2.3).

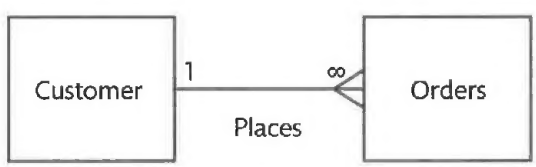


Figure 2.3: An ERD showing the relationship between the Customer and Orders tables

Note that the connecting line which shows the relationship has a '1' at the customer end and a ∞ at the order end. This shows that the Orders table can have many records related to one record on the Customer table.

Although the one-to-many entity relationship is the most common, there two other types of entity relationships which can exist.

One-to-one entity relationships

In a one-to-one entity relationship, one record in the first table is related to one and only one record in the other table. An example of this might be an employee database. Some employees might have a company car or van. Data about company cars is likely to be kept on a separate table (the Company car table) from general employee data (the Employee table), because it is not data about the employee directly and not every employee will have a company car or van (only those who need to travel to complete their work). However, no employee can have more than one company car, so the relationship between the Employee table and the Company car table is a one-to-one relationship. Every record on the Company car table is related to one and only one record on the Employee table. Similarly, one and only one record on the Employee table is related to one and only one record in the Company car table. When drawn in an ERD diagram, the relationship looks as shown in Figure 2.4.

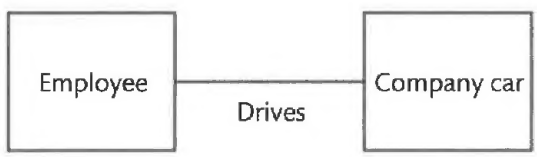


Figure 2.4: An ERD diagram showing the one-to-one entity relationship between the Employee and Company car tables

Many-to-many entity relationships

The final type of relationship is the many-to-many entity relationship where one record in the first table can be related to many records in the second table, and *vice versa*. So, unlike with a one-to-many relationship, one record in the second table can also be related to many records in the first table. Imagine a database of college students and the subjects that they study. In the Student table, each student record will be related to many subjects (listed in the Subject table) that each student studies. However, in the Subject table, each subject record would be related to the many different students that take that subject (listed in the Student table).

Many-to-many relationships cannot be represented directly in a relational database and have to be broken down into three tables with a link or mapping table linking the two main tables. The example of the Student and Subject tables can be used to explain how this works. As already mentioned, linked tables require a unique identifier for each record so, in this example, the unique identifier for the Student table is likely to be a student

number, while for the Subject table it is likely to be a subject code. To make the mapping table (which can be called Student-subjects) the unique identifier attribute is taken from each table and used to create a link between students and subjects. Where a single student is, for example, taking four subjects, four records are created in the Student-subjects table, all with the same student number but with each record containing a different subject code for the four subjects they are studying. See Figure 2.5 for an ERD diagram for the many-to-many relationship between students and subjects.

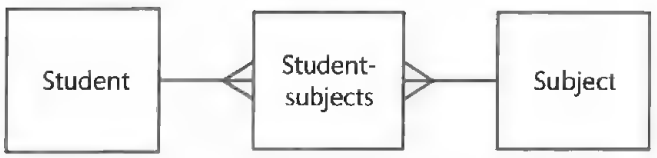


Figure 2.5: An ERD diagram showing the relationship between the Student and Subjects tables using a mapping table called Student-subjects

In addition, the Student-subjects mapping table has a one-to-many relationship with the Student table (one student can be related to many mapping table records) and, likewise, the Subject table has a one-to-many relationship with the Student-subjects table, so one subject record is related to many records on the Student-subjects table.

Generic and semantic

Relationships can be modelled in a *generic* way (that is, generalised and non-specific) when you first attempt to understand the design of a complex system. Then, as your understanding develops, you can create a *semantic* (that is logical) model of the relationship between specific data items. The topic of designing database relations is covered in more detail under section B, when conceptual and logical data models are explained.

Link

For more on designing database relations see Relational database design.

Relational keys

We have already mentioned the need for unique identifiers, such as a customer number or subject code, in the tables of a relational database. These unique identifiers are known as keys. Keys not only uniquely identify records within a database, but they can also be used to speed up searching for records because the database keeps an index of keys so that it can find key values without searching through the whole database.

Primary key

The unique identifier for a table is known as the **primary key**. Each table should have a primary key and, within the table, each key value must be unique.

Candidate keys

When designing a database table you may have a choice of fields which may make a suitable primary key.
For example, a company may have a database of its employees. Possible primary keys that they could use might be the employees' national insurance (NI) numbers or unique employee numbers. These different possible primary keys are known as the **candidate keys**. During the database design process, the decision must be made as to which of the candidate keys is to be chosen as the primary key, as there can only be one primary key per table in a database.

Foreign keys

As already explained, when creating a relationship between two tables, the primary key value of one table is inserted into the table that it is linked to. So, using the example discussed earlier of the online ordering system, orders are related to the customer that ordered them by taking the customer number (the primary key) from the Customer table and putting it into the orders saved on the Orders table, which shows the orders that the particular customer has placed. This value (customer number) could not be the primary key value on the Orders table because a customer can place many orders, so it would not be unique in the Orders table. Instead, the Orders table would most likely have a primary key of order number. However, the customer number in the Orders table is known as a **foreign key** because it is a key in another table.

Key term

Primary key – a field which can uniquely identify one and only one record in a table.

Candidate key – a field or combination of fields which could provide a unique identifier for a table. Typically in a table there may be several candidate keys which could be chosen as the primary key for the table.

Foreign key – the primary key from a record in a table at the 'one' end of a one-to-many entity relationship, which is used as a field in records in the table at the 'many' end, to provide the link between the records.

Composite keys

In some situations it may be necessary to combine several attributes to create a unique key for a table. This is known as the **composite key** (also called a super key). For example, imagine a database of books in a library. The book title alone would not necessarily uniquely identify a book, as two books could have the same title. However, the book title combined with the author's name might be a better choice. But suppose there were several editions of the book. In this case, the book title and author name would not be unique, but including the edition number in the composite key as well would create a unique value (a super key). There might, of course, be several copies of the book in the library, in which case the copy number would also need to be included in the super key.

Key term

Composite key – the combination of fields which can uniquely identify each record in a table (also known as a super key).

Integrity constraints

The integrity of the data in a database is important. For example, in the online ordering system, what would happen if there was a record in the Orders table with no related record on the Customer table, perhaps due to an invalid customer number? This would be a 'lost' order with no customer to deliver it to. The issues of entity and referential integrity are covered in detail later in this unit.

Manipulating data structures and data in relational databases

Having data organised into related tables is all very well but, for the database to be useful, there must be a way to find and extract data from the database. There also need to be methods for giving users access to the database, and for securing and recovering the database.

We have already used the tools that Access® provides to create tables. Once a table has been created, data can be entered into the table using the Datasheet View. You can also modify data in the table in Datasheet View. You can return to the Design View of the table at any time and modify its structure, although care should be taken in making changes to fields which already contain data. For example, if you change the datatype of a field from text to a numeric datatype, then data already in the field will not match the new datatype. You can also delete records in Datasheet View.

Typically, in a database containing a large number of records, you would want to be able to find a specific record (or group of records) within a table which match certain requirements. This can be done using the standard database query language known as SQL, which can be used on almost any relational database, not just Microsoft® Access®.

SQL

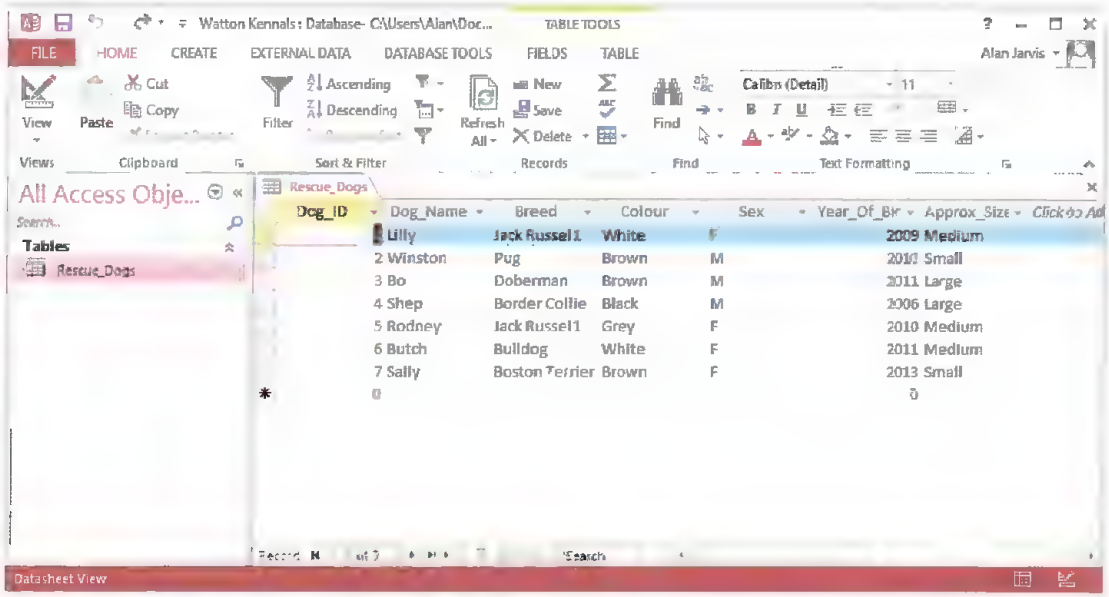
Structured query language (SQL) is the standard language used to create queries and extract data from a relational database. SQL is used to define, modify and remove data structures and data, including allowing you to insert new records into a database, delete records and update records.

Retrieval of data for queries and reports

SQL can be used to retrieve data for queries and reports using the SQL command select query.

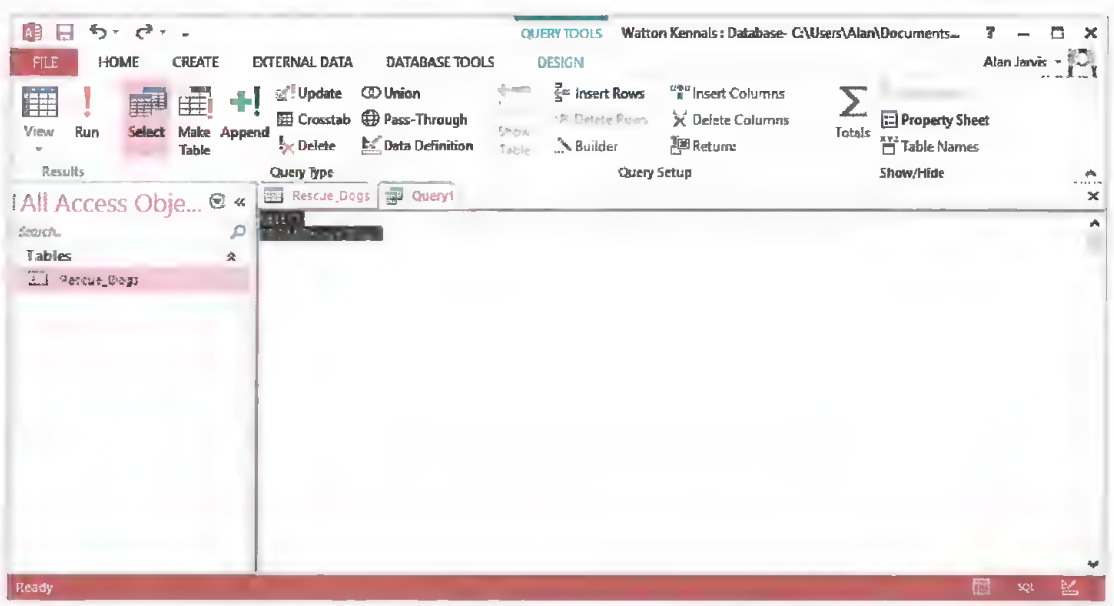
Step by step: SQL Select query

1 To demonstrate how to use SQL commands to retrieve data, we will use a simple single table Access® database that might be used by an animal charity to keep records for rescue dogs awaiting adoption. The single table (called Rescue_Dogs) is shown here.



2 To create a query in Access®, click on Create in the menu bar, then chose Query Design in the toolbar. Access® will ask you which table you want to add to the query: there is only one, Rescue_Dogs, so click Add, and then Close to close the Show Table dialog box.

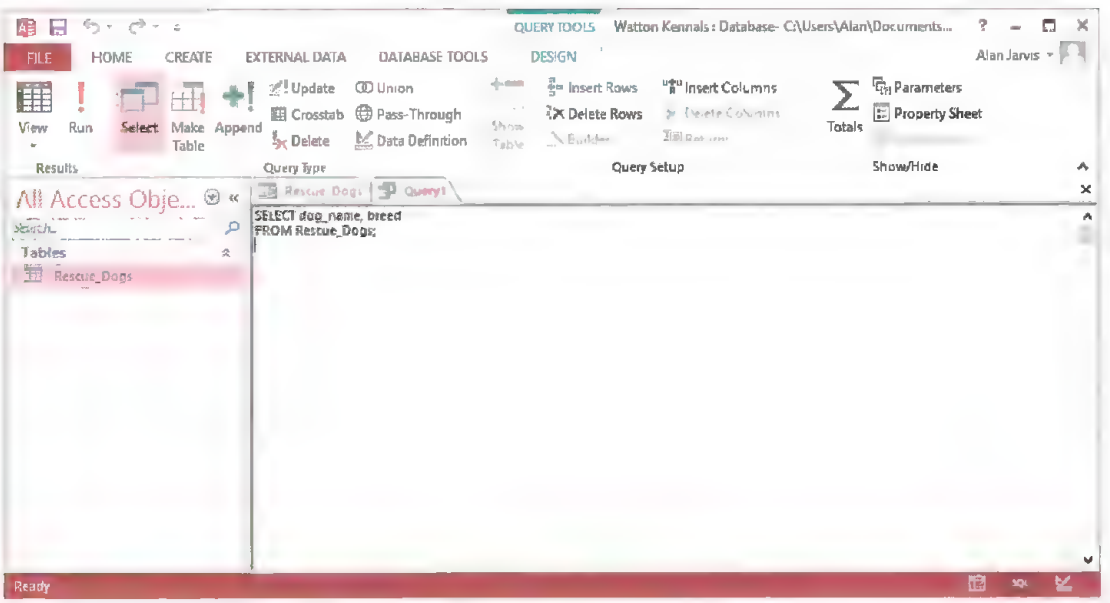
The default view that Access® provides for creating queries is the Design View, which provides a visual drag and drop interface for creating queries. However, you want to use SQL View, so click on the View icon (far left of the toolbar) and choose SQL View from the drop-down list. You will then see the display shown here.



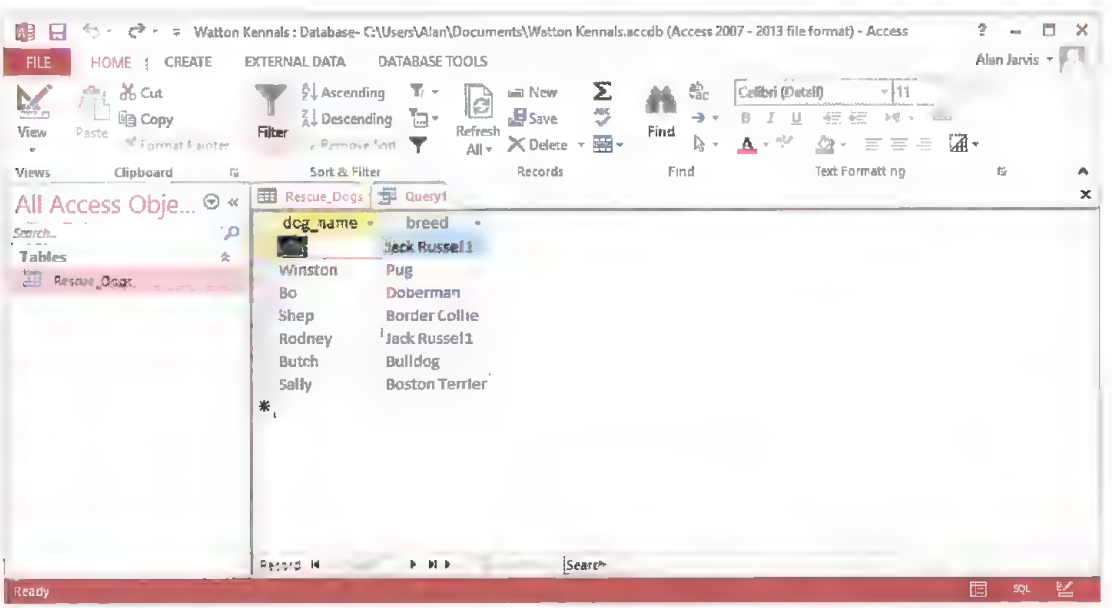
3 One of the simplest things you can do with the SQL Select statement is just display certain fields. Suppose you just wanted to display the dog name and breed. The SQL statement for that is:

```
SELECT dog_name, breed FROM Rescue_Dogs
```

You can see this in SQL View here.



4 Now click the View icon again and select Datasheet View. You should see something like this.



Using SQL to select fields to display is useful, but what is even more useful is the ability to select records from the database which match certain criteria. This can be done using the SQL SELECT...WHERE clause.

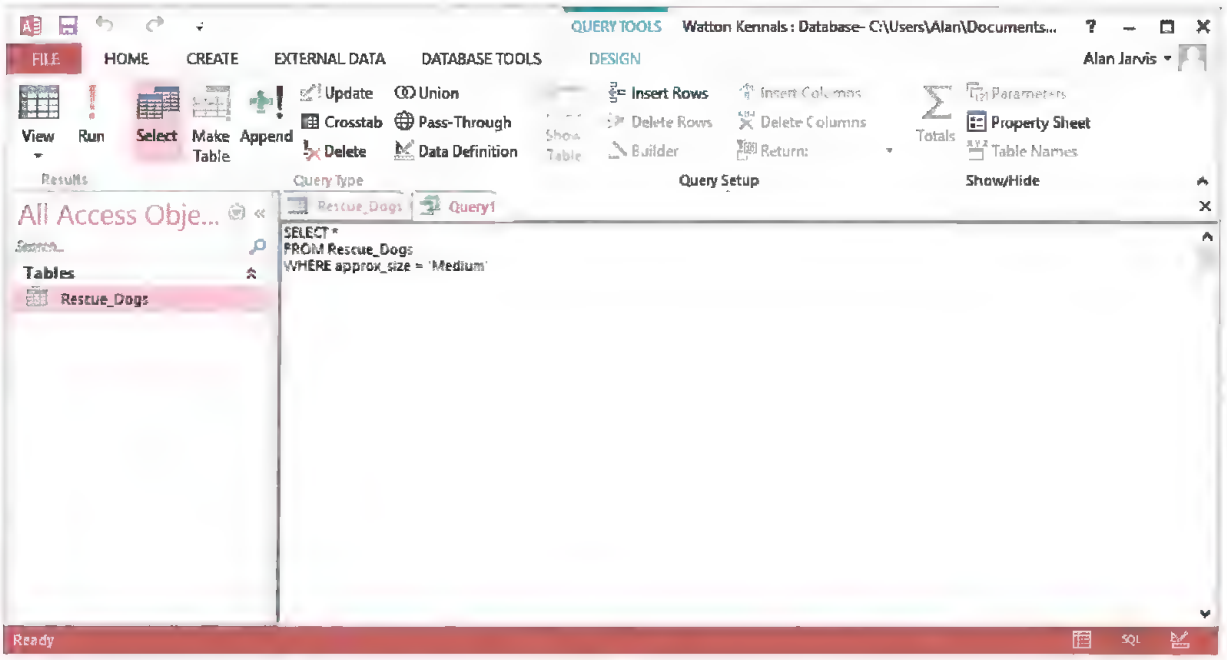
Step by step: SQL Select...where

2 Steps

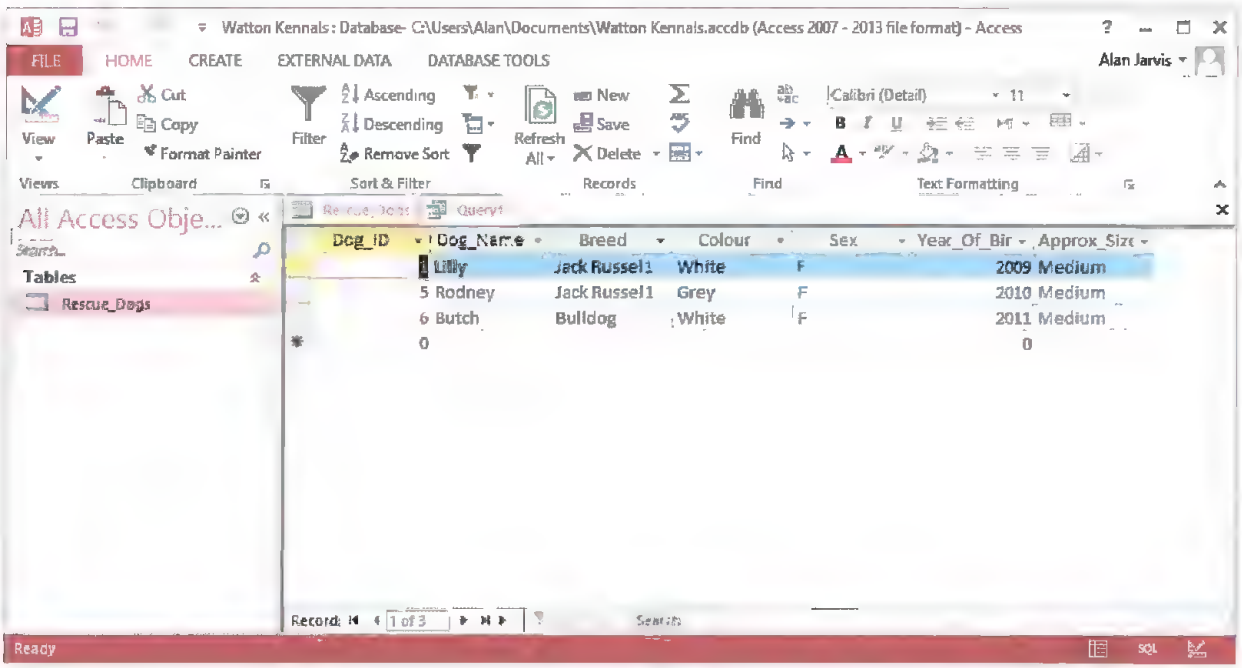
1 Suppose, for example, you were only interested in medium size dogs. This kind of selection uses the SELECT... WHERE clause. To display all the fields you use the * symbol, so the command would be:

```
SELECT * FROM Rescue_Dogs WHERE approx_size = 'Medium'
```

This is shown here.



2 If you swap from the Query View to the Datasheet View, you can see the result of the SQL command, with just medium dogs listed, as shown here.



Inserting

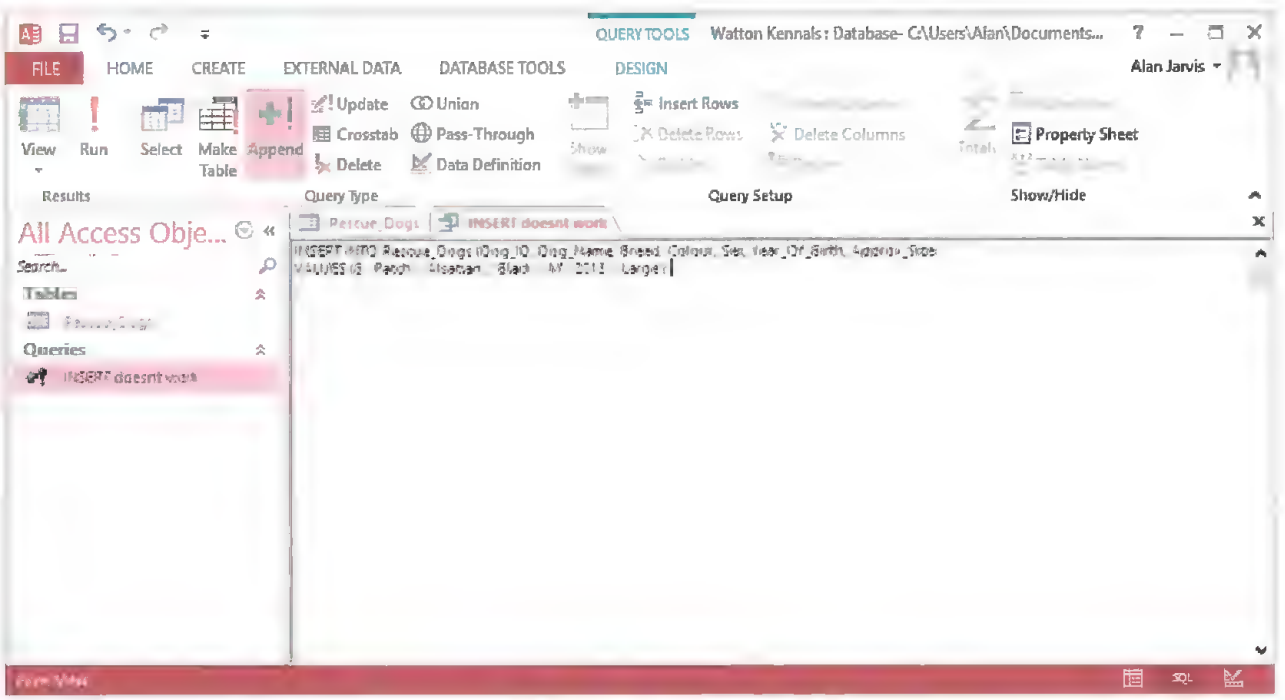
You can also use SQL to insert new records. This is done using the SQL command INSERT INTO.

Step by step: SQL Insert into

1 Change the type of query to an Append Query by clicking the Append icon in the toolbar. Access® will ask you where you want to append the records to, so select the only table: Rescue_Dogs. The correct command to insert a single record is as follows:

```
INSERT INTO Rescue_Dogs (Dog_ID, Dog_Name, Breed, Colour, Sex, Year_Of_Birth, Approx_Size)
VALUES (8, 'Patch', 'Alsatian', 'Black', 'M', 2013, 'Large')
```

(Note that the text fields are enclosed in quotes but the numeric fields are not.)
The command is shown typed into SQL View here.



2 To insert (append) the record, click the Run icon in the toolbar. Access® warns you that you are about to append a record. Click Yes. This will then insert a new record into the table.

Updating

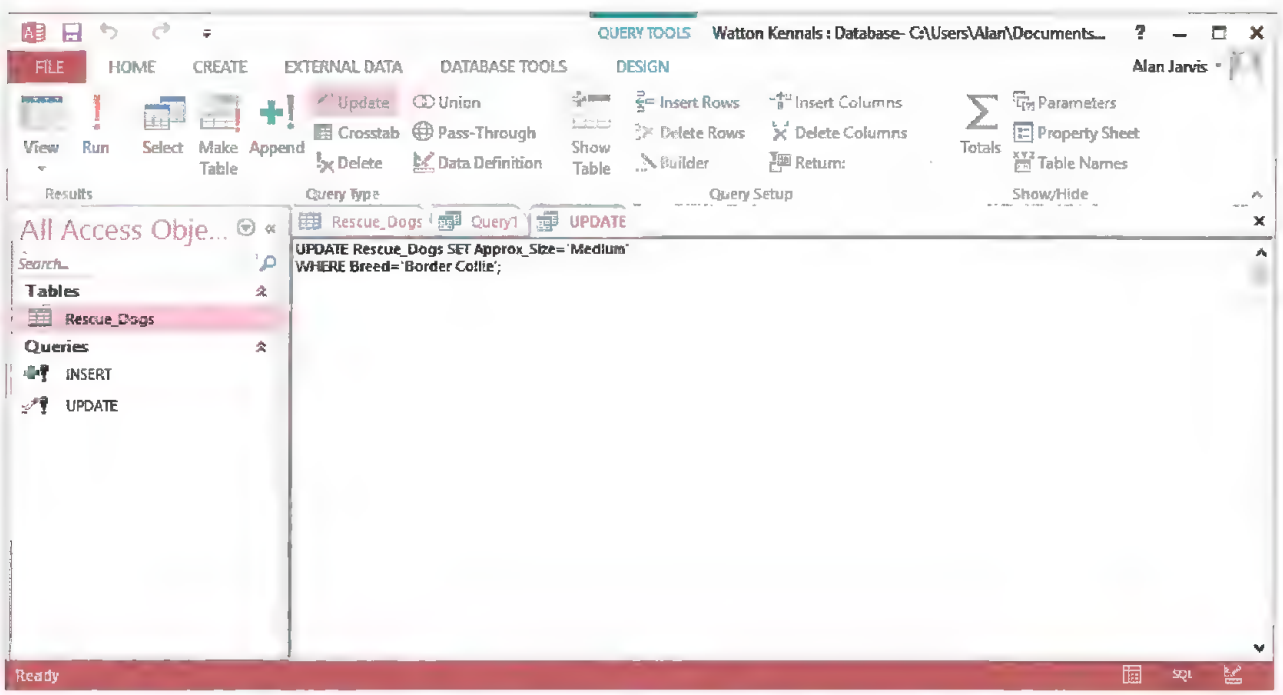
The final SQL command that you will need is the Update command. As the name suggests, this allows you to update existing records.

Step by step: SQL Update

1 To create the command, first use the Create menu to make a new Query Design. Add the Rescue_Dogs table, as before, then click the Update icon in the toolbar to make it an update query. Now swap to SQL View. Suppose it was discovered that a mistake had been made and the Border Collie breed of dogs should have been classified as 'Medium' size rather than 'Large'. The SQL command to update the data accordingly is:

```
UPDATE Rescue_Dogs SET Approx_Size='Medium'
WHERE Breed='Border Collie'
```

This can be seen here.



2 Click the Run icon and Access® warns you that you are about to update a record. Click Yes and the update is completed.

3 You can also delete a record using SQL. The command is used in a similar way to the select command, so the command would delete all the medium sized dogs from the table. Care should be taken when using delete. There is no 'undo' option.

```
DELETE FROM Rescue_Dogs WHERE approx_size = 'Medium'
```

Using SQL like this seems very longwinded compared with using the drag-and-drop query design interface. However, you would not normally use SQL like this – it would be built into a database procedure which runs automatically without user intervention.

The output from SQL statements or queries can be displayed in the table format shown here, but more commonly reports or forms are used to format and display the output.

Link

There is a lot more you can do with SQL. You can find a full SQL course with examples at www.w3schools.com.

Administration of users and database security

Where a database is used by multiple people, that is, a multi-user database, it is important that there are procedures in place to control what actions different users can carry out on the database. This is because not all users need to be able to edit and they may only want to view the database. It can also be safer to not allow editing access to all users so that human error is less likely to occur. It is unlikely that all users would be allowed full access to every part of the database.

Controlling which actions can be carried out by different users is typically done by creating users and then granting them permissions, such as the ability to insert, update and delete records. The primary method of maintaining database security is by maintaining the system of database permissions, as described above, by ensuring that users have strong passwords and that they only have the permissions required to do their jobs and no more.

In addition, the following security procedures can help to secure a database.

- Hardening of the underlying system by the application of standard security procedures: ensuring the system has up-to-date software, virus protection, and through the use of firewalls.
- Database auditing and monitoring can be used with sensitive data. Logs are kept of the changes to the database and are reviewed regularly. Database monitoring systems can automatically look for unusual activity on the database in an attempt to identify things such as credit card fraud or an attack by a hacker.

Integrity and recovery

Ensuring the integrity of a database is a very important consideration and there are two main aspects to it: design integrity and physical integrity.

Link

The topic of design integrity is covered in the section on Database anomalies.

The design of a database needs to ensure that there are no inconsistencies in the structure of the tables (entities) and the ways in which they are related: this is entity integrity. The process of normalisation, as described in the next section, is used to check the entity integrity of the database.

Physical integrity

There must be mechanisms in place to ensure that data stored on disk is protected from corruption: this is physical integrity.

Multi-user server-based database systems are usually implemented on high availability hardware. These include RAID disk systems, which use multiple disks to provide protection from single disk hardware failures, and other duplicated hardware. They provide a good degree of protection from hardware failure. However, the system can still be vulnerable to software failure. If the database system crashes part way through a **database transaction**, this can lead to inconsistencies and a mechanism to recover the database is required. Depending on the application, certain types of transaction must be completed, either in their entirety or not at all.

Key term

Database transaction – A complete set of actions required to complete some task in a database. Transactions related to database systems, such as withdrawing money from an ATM machine or booking an airline ticket, are made up of several steps such as authentication of the user, selecting the required item and paying for a purchase.

Many database transactions consist of data being added or updated on several tables. For example, when booking an airline ticket, there are several parts to the transaction, such as reserving the seat and paying for the ticket. In order to ensure the integrity of the transactions, database systems keep a log of changes to the database which have been completed and commit points where transactions have been successfully completed. Should the system crash, the log is used to identify any transactions which have not reached the commit point (that is, they have not been successfully completed). Those changes which are part of an incomplete transaction are then undone or rolled back, to the previous commit point. This ensures that the database is recovered to a consistent state.

PAUSE POINT

What kinds of database need strong security? From time to time, commercial databases are attacked and you may read in the news about the problems they have encountered. Have you heard of any recent attacks? How was data stolen and what were the consequences?

SQL injection is a software vulnerability which can be used by hackers to gain access to data held on databases associated with web applications. SQL injection has been used in a number of recent high profile attacks in which large quantities of data have been stolen.

What is SQL injection and how does it work?

How can you protect against it?

Normalisation

When considering a real-life situation that you want to model using a relational database, it can sometimes be difficult to see how to divide up the data into different related tables in the most efficient way. The process of normalisation has been developed to help database designers create designs that avoid data redundancy (duplicated data), which, in turn, minimises storage requirements and helps to avoid the problems that occur with duplicated data. For example, if a customer's postcode is held on a database in two different tables, then should the customer move house and change their postcode, it may not be updated in more than one table, which may cause anomalies.

The process of normalisation is an important part of the database design process. The outcome of the process of normalisation will be a **verified** database design which identifies the main structure of the database, including:

- the tables that the database will be split into
- the indexes used, including the primary and foreign keys and any composite keys required for the tables
- the structure of the individual tables from which the data dictionary can be created.

Key term

Verification – Verification is the process of checking that something is correct. Normalisation is the process used to verify the database design.

Following on from the normalisation process, database designers need to consider the issue of referential integrity and how the updating and deletion of records on related tables is dealt with.

The stages of normalisation

Normalisation is a three-stage process which begins with raw data, such as you might extract from a paper-based system that the database you are designing will replace.

Summary of the stages of normalisation

1. The process starts off with raw data, which is referred to as un-normalised (UNF).
2. The first stage takes raw data and modifies its structure to the meet the requirements of 1st normal form (1NF).
3. The second stage further modifies the 1NF structure to meet the requirements of 2nd normal form (2NF).
4. The third stage takes the 2NF data structure and modifies it to meet the requirements of 3rd normal form (3NF).

To demonstrate the process we will return to the example of the online ordering system. Figure 2.6 gives an idea of the sort of paper-based order form the database will emulate.

This is our un-normalised data (UNF). Note that the order number and the customer details appear just once but the product details repeat for each item ordered.

Universal Products Ltd

Order No 1

Customer: Bert Smith
10 High Street
Norwich

Ordered Items

Product ID	Product description	Price	Quantity	Total
P33	Camera	£35.00	1	£35.00
P09	Book	£8.00	3	£24.00
P21	Headphones	£22.00	2	£44.00
			Total	£103.00

Figure 2.6: Paper-based order form

Stage 1: 1st Normal form

The first stage of normalisation involves modifying the data to meet the requirements of 1st normal form (1NF) by doing the following.

- Remove any calculated fields (such as total order value).
- Make sure that each data item is **atomic**. The customer name is not atomic since it can be split into 'First name' and 'Surname'. The same is true of the customer address but, for simplicity, we will ignore this attribute.
- Remove repeating groups so that there is a data item for every record for each attribute. So for example, in the paper-based order form, the order number and customer number occur just once but we must include them in each record on our new 1NF table.
- Finally, we must select a primary key (which needs to be unique). There is no one value which is unique but we can create a composite key using the Order Number and the Product ID.

Key term

Atomic – in this context, atomic means broken down into individual parts. For example, in terms of its suitability for a database field a person's name is not atomic because it can be broken down into title, first name, middle names and surname.

The completed 1NF table is shown here (Table 2.10). More orders have been added to make the process clearer. The key fields are indicated by underlining the attribute name.

Table 2.10: Completed 1NF table (underlined attributes are the key fields)

Order no.	Product ID	Customer no.	Customer FName	Customer SName	Description	Price	Quantity
1	P33	C10	Bert	Smith	Camera	£35	1
1	P09	C10	Bert	Smith	Book	£8	3
1	P21	C10	Bert	Smith	Headphones	£22	2
2	P09	C5	Wendy	Jones	Book	£8	1
2	P15	C5	Wendy	Jones	Trainers	£40	1
3	P21	C12	Alice	Williams	Headphones	£22	1
3	P19	C12	Alice	Williams	Pen	£6	2

Stage 2: 2nd Normal form

The second stage of normalisation is to modify the data to meet the requirements of 2nd normal form (2NF). If a table has a primary key based on a single attribute, then

nothing needs to be done at the 2NF stage. However, our table does have a composite key. In this case, we need to move any data that is only **dependent on** one part of the composite key attributes to a separate table.

The data that are only dependent on one part of the composite key attributes are the Description and Price, as these only depend on the Product ID and should not be in the OrderNumber table. If you know the Product ID, then you can find the Description and Price, but knowing the order number alone will not uniquely identify the product (as several items may be on the order). The completed 2NF tables now look like this (Tables 2.11–12).

Key term

Dependent on – one field depends on another if you can only find out the unique value of the second field if you know the value of the first one. For example, if you know the Product ID, then you can find the description and the price.

Table 2.11: Completed 2NF – Product table

Product ID	Description	Price
P33	Camera	£35
P09	Book	£8
P21	Headphones	£22
P15	Trainers	£40
P19	Pen	£6

Table 2.12: Completed 2NF – Main table

Order no.	Customer no.	Customer FName	Customer SName	Product ID	Quantity
1	C10	Bert	Smith	P33	1
1	C10	Bert	Smith	P09	3
1	C10	Bert	Smith	P21	2
2	C5	Wendy	Jones	P09	1
2	C5	Wendy	Jones	P15	1
3	C12	Alice	Williams	P21	1
3	C12	Alice	Williams	P19	2

Note that the duplicates are removed from the Product table but the Product ID is left in the main table as a foreign key.

Stage 3: 3rd Normal form

The third and final stage in normalisation is to modify the data to meet the requirements of 3rd normal form (3NF). The requirement of 3NF is that none of the non-primary key attributes should depend on any other attributes. If they do, then they need to be moved to another table, leaving a copy of the primary key field for the new table in the original table as a foreign key.

In the main table, Customer no. will allow you to find a unique customer name (First and Surname) and, since these are not key attributes, they need to be moved to another table (the Customer table) with a copy of the key for that new table (customer number) left in the main table as a foreign key. The 3NF tables are shown here (Tables 2.13–15).

Table 2.13: 3NF – Product table

Product ID	Description	Price
P33	Camera	£35
P09	Book	£8
P21	Headphones	£22
P15	Trainers	£40
P19	Pen	£6

Table 2.14: 3NF – Main table

Order no.	Customer no.	Product ID	Quantity
1	C10	P33	1
1	C10	P09	3
1	C10	P21	2
2	C5	P09	1
2	C5	P15	1
3	C12	P21	1
3	C12	P19	2

Table 2.15: 3NF – Customer table

Customer no.	Customer FName	Customer SName
C10	Bert	Smith
C5	Wendy	Jones
C12	Alice	Williams

However, the data shown in Tables 2.13–15 are not fully normalised yet because in the main table there is dependence between Order number and Customer number. Since an order is only placed by one customer, if you know the order number, you can find the customer number. Order number and customer number therefore need to go in their own table (the Orders table) as shown here in the completed 3NF tables (Tables 2.16–19).

Table 2.16: Completed 3NF – Product table

Product ID	Description	Price
P33	Camera	£35
P09	Book	£8
P21	Headphones	£22
P15	Trainers	£40
P19	Pen	£6

Table 2.17: Completed 3NF – Customer table

Customer no.	Customer FName	Customer SName
C10	Bert	Smith
C5	Wendy	Jones
C12	Alice	Williams

Table 2.18: Completed 3NF – Orders table

Order no.	Customer no.
1	C10
2	C5
3	C12

Table 2.19: Completed 3NF – Order products table

Order no.	Product ID	Quantity
1	P33	1
1	P09	3
1	P21	2
2	P09	1
2	P15	1
3	P21	1
3	P19	2

Note that the remaining table is named the Order products table since it identifies which products are on each order.

Research

Although for this qualification you only need to understand the process of normalisation up to 3rd Normal form (3NF), there are higher levels of normalisation. Find out what these are and why they might be needed.

Database anomalies

The process of normalisation is designed to help reduce the possibility of anomalies occurring as the database is used. To see how anomalies might occur in a database which has not been fully normalised we shall look at an expanded version of the table we used to demonstrate how data at 1NF looked in the section Stage 1 of The stages of normalisation (see Table 2.20).

Table 2.20: Expanded version of Completed 1NF table

Order no.	Product ID	Customer no.	Customer FName	Customer SName	Address	Description	Price	Quantity
1	P33	C10	Bert	Smith	10 High St	Camera	£35	1
1	P09	C10	Bert	Smith	10 High St	Book	£8	3
1	P21	C10	Bert	Smith	10 High St	Headphones	£22	2
2	P09	C5	Wendy	Jones	3 Station Rd	Book	£8	1
2	P15	C5	Wendy	Jones	3 Station Rd	Trainers	£40	1
3	P21	C12	Alice	Williams	2 London Rd	Headphones	£22	1
3	P19	C12	Alice	Williams	2 London Rd	Pen	£6	2

The following type of anomalies can occur with this kind of 1NF database table:

- insert
- delete
- update.

Link

See the original table of data at 1NF in the section Stage 1 of The stages of normalisation Table 2.10

Insert anomalies

To insert a new order into the database, the customer's name has to been entered even if that customer has previously placed an order. This gives rise to the possibility of errors or inconsistencies.

Another possible source of anomalies is the fact that an existing customer might need to enter their address every time they place an order. Compare this with the 3NF version of the database (Table 2.17). In this, the customer details (only Customer Fname and Sname are shown, but address details would be in the same table) are recorded once only per customer in the Customer table. Only the Customer no. attribute needs to be recorded in the Orders table.

Another issue would be registering of new customers. In the 1NF version of the database, a new customer could only be registered if they placed an order as the Product ID is part of the composite key. Therefore, it is not possible to create a record with a null value in the Product ID attribute (that is, it is impossible to create a customer account without placing a record).

Delete anomalies

If a customer places an order and then later cancels it, it needs to be deleted. However, if that customer does not have any other orders, then all their details will be lost since without an order you cannot have any customer details (you cannot have a customer record). This is avoided in the 3NF version of the database as customer details are held in a separate table.

Update anomalies

If a customer changes their address, for example, then all the records containing orders for that customer may need to be searched to change the address on those orders too. Again, this is avoided in the 3NF version of the database, as customer details are held in a separate table.

Another issue which makes the 1NF version of the database unworkable is that of products and their details. In this version, only products which have been ordered can exist on the database. The only way to add a new product is if someone orders it. This clearly would not be acceptable in practice.

As you can see from the normalisation example we have completed, the process helps you to decide on the indexing, including the primary, foreign and composite keys required for each table. The completed table diagrams can be used as the basis for the data dictionary, which will include full details of the tables, attributes, data types and validation required for the whole database.

Referential integrity and cascading update/delete

One issue that can occur with a relational database is how to deal with linked records when deleting or updating a record at the 'one' end of a one-to-many relationship. Consider the Customer and Orders relationship. One customer can place many orders and the link is made by placing the Customer table primary key attribute as a foreign key in the Order table.

But what happens if a customer closes his account with the company? If we delete the customer record alone, that will leave orders on the Order table with no corresponding customer. These might be considered 'lost' orders and might cause problems with an application which expects to see every order linked to a customer record. One alternative is to use cascading updates, where deleting a record on the Customer table would automatically delete related records on the Order table. Microsoft® Access® (and SQL Server) provides an option to switch this facility on, which is known as referential integrity.

Reflect

In reality, would you want to completely delete a customer record and their associated orders?

What alternatives might there be?

Are there any legal implications of retaining customer details for people who close their accounts?

Another problem may occur if you were to allow updates to the key field to which other records are related. Imagine an employee database where the key field for the Employee table was the employee's NI number. Staff annual appraisal records are linked to the employee record by inserting the primary key from the Employee table (the NI number) as a foreign key on the Appraisals table. Consider what would happen if it was discovered that a data entry error had been made in one employee's NI number and it was updated to be correct. This would mean that all of the employee's appraisal records would be 'lost' because the NI number on the parent record had changed. The referential integrity setting in Microsoft® Access® and SQL Server allows you to prevent changes to key fields which exist as foreign keys in other tables.

Reflect

Is the use of an employee's NI number as the primary key field a good idea?

What alternative key fields could be used to avoid the problem of updates?

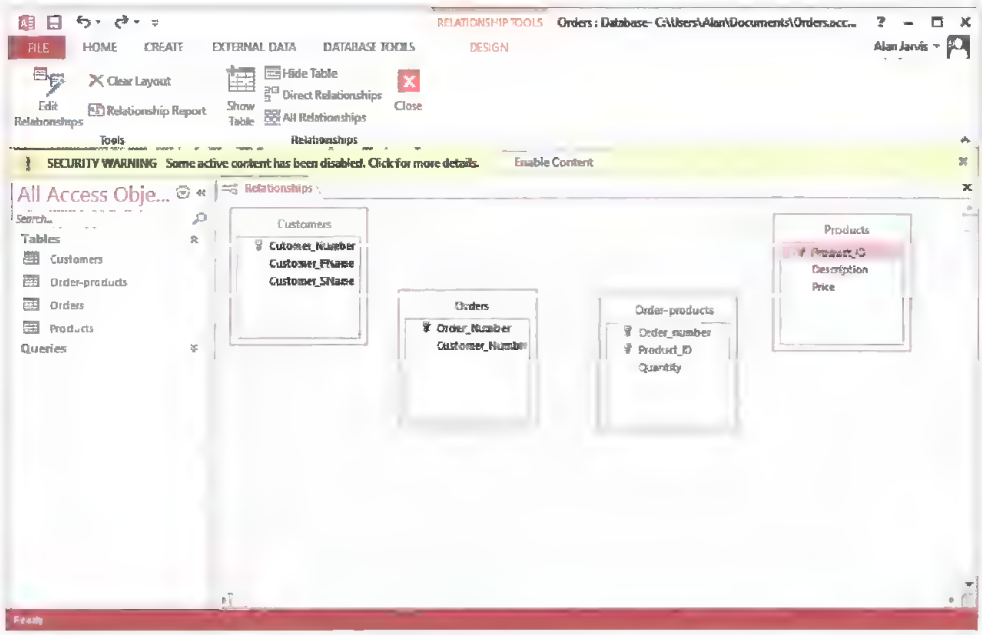
Step by step: Creating relationships in Access®

6 Steps

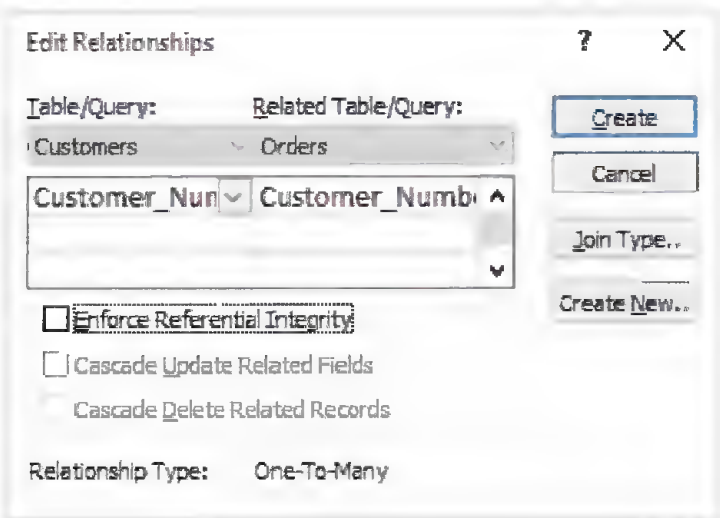
1. You can create as many tables as you like in a single Access® database. Linking them together can be done as long as your design is correct and you have used the primary key from the 'one' end of a relationship as a foreign key in the table at the 'many' end of the relationship.

2. Assume that, as well as the Customer table that we created earlier, you have also created tables for Products, Orders and Order-Products following the design created during the normalisation process described in the Stages of normalisation.

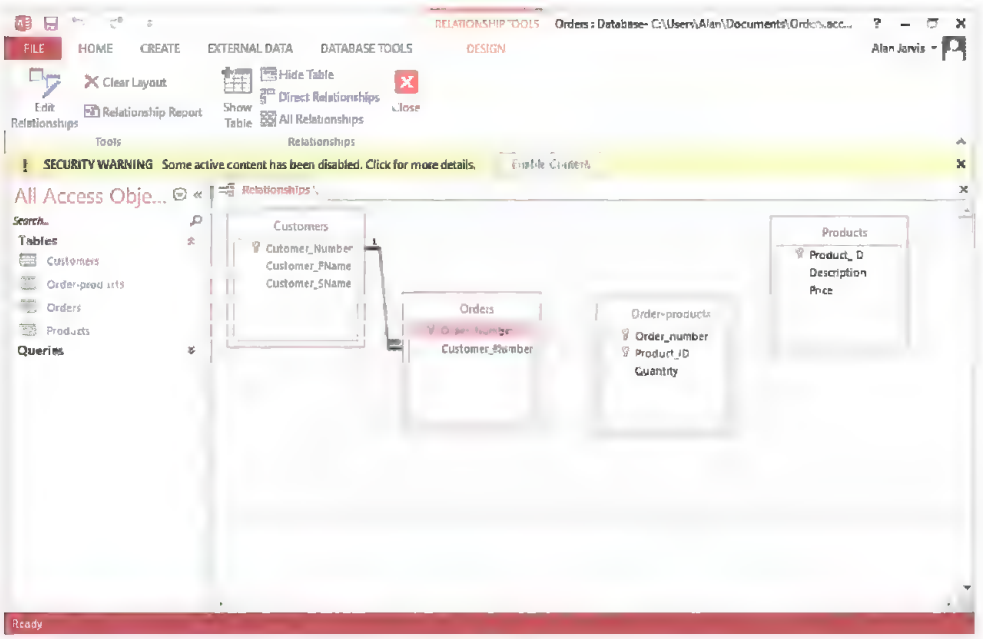
To link them together, go to the Database Tools menu in Access® and click on the Relationships icon in the toolbar. The Show Table dialog box will appear, asking which tables you want to create relationships for. Click each of the tables in turn and click Add then, after you have added them all, click Close. The relationships display should now look like this.



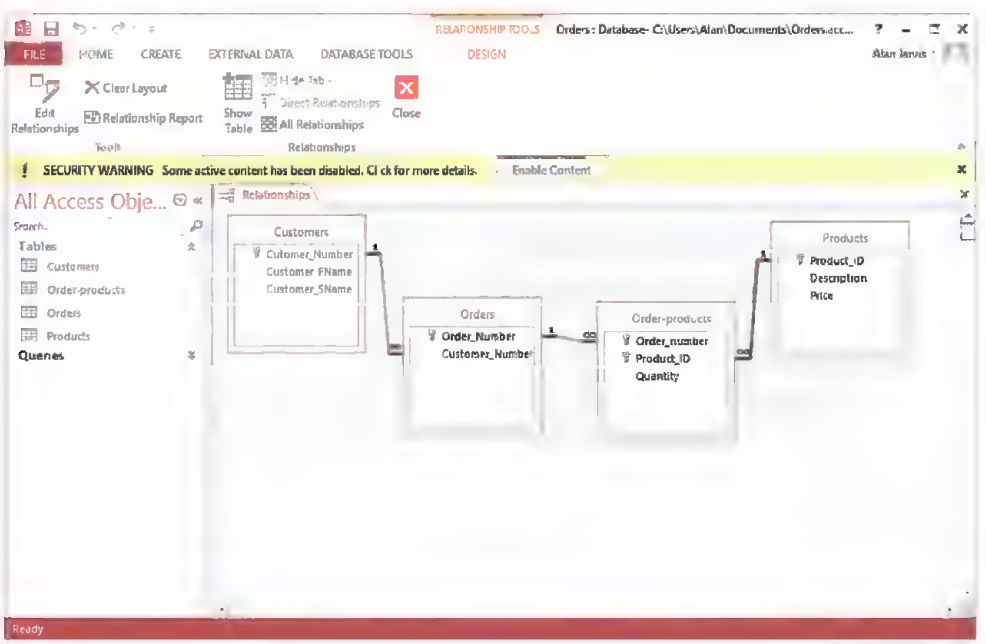
First, we will create the relationship between the Customer table and the Orders table. Click on the Customer_Number field in the Customer table (the primary key) and drag it over to the Customer_Number field in the Orders table (the foreign key). The dialog box shown here will appear.



The appearance of the dialog box shows that Access® has identified that this is a one-to-many relationship (it knows this because the field is a primary key on the Customer table). It also gives you the option to enforce referential integrity so that it will not allow you to delete records or modify the primary key value of records on the Customer table that have related orders. You can also turn on the Cascade Update and Cascade Delete features so that if a customer is deleted from the customer table all of their orders will automatically be deleted (rather than you just getting an error message), or if you update the Customer number field on the Customer table related records on the orders will also be updated. Turn these options on and then click Create. The relationships display should now look like this.



Now do the same with Order_Number in the Orders table (primary key) and the Order_number in the Order-products table. Finally create the relationship between the Product_ID (primary key) in the Products table and the Product_ID in the Order-products table. The relationships should now look like this.



B Standard methods and techniques to design relational database solutions

Now that you have looked at the theoretical structure of RDBMS, you will investigate the standard methods and techniques used to design relational database solutions. As with any software application development, the process starts with building an understanding of the user requirements and progresses through to creating more and more detailed models and designs for the system.

Relational database design

In many cases, databases can be very large and complex and a lot of design work needs to go into their creation and development. The design needs to be carefully considered, as resolving design errors during the development or implementation of the database can be difficult and costly. As well as the designs for the application, the selection of suitable hardware, software tools, techniques and processes needs to be considered. With a multi-user server database, hardware needs to be selected which has sufficient performance and capacity to support the expected transaction volumes.

Database design

The designing of a large and complex database system involves a number of phases. The first phase is a broad conceptual overview of the database design, after which more detail and complexity are added in later design phases. The design process is an iterative one, with continual refinements and improvements being made. There are often three phases to the design process.

1 Conceptual database design – This is the first phase and involves building a **conceptual model** of the information used in the database system based on the details provided in the user requirements specification. It does not take into account the physical implementation details such as the database software to be used or hardware platforms. It involves the identification of tables and their relationships, as well as the drawing of an outline ERD.

2 Logical database design – This phase takes the conceptual model and refines it further into a **logical model**. The process of logical design involves arranging data into a series of logical relationships (tables and fields) and taking into account the requirement of the relational database model and the selected RDBMS software. The logical design defines the detailed structure for the tables, relationships and fields. This design is then validated using the normalisation process.

3 Physical database design – This final phase involves deciding how the database will actually be implemented on the chosen RDBMS software and hardware. It includes calculations of the required storage space. It defines the storage structures, methods of access that will be used and security protection that will be applied.

Key term

Conceptual model – is based on generalisation and ideas (concepts) such as 'a customer places an order'.

Logical model – takes the concepts and applies rules (logic) to the concepts (including verifying them with normalisation) and adding details such as the nature of the relationship which exists between customer and order (via the primary and foreign keys).

The table below (Table 2.21) shows at which phase various aspects of the design are defined.

Table 2.21: Phases at which design aspects of a database are defined

Design feature	Design phase		
	Conceptual	Logical	Physical
Entity names	✓		
Entity relationships	✓		
Fields		✓	
Keys (primary and foreign)		✓	
Field names			✓
Field datatypes			✓

Link

An example of conceptual and logical design models can be found in the section on Design documentation.

Relational algebra

As part of the design process, the types of relationship between tables will be defined using entity-relationship (ER) models and tested using normalisation. Most

relationships are likely to be one-to-many, with the primary and foreign keys identified as part of the design process. There may also be some one-to-one relationships required. Remember that many-to-many relationships, identified in the conceptual design phase, will need to be split into separate linked tables at the logical design phase.

Link

For a reminder of the various different types of entity relationships, see Database relations.

The queries used to extract data from the database also need to be designed, using SQL and Boolean operators such as AND, OR and NOT and comparison operators such as > (greater than), < (less than), ≥ (greater than or equal to), ≤ (less than or equal to).

Link

For more on Boolean operators, see *Unit 5: Data modelling, Logical functions*, and for more on comparison operators, see *Unit 7: Mobile apps development, Operators*.

RDBMS and SQL selection

There are a range of different RDBMS software applications available, almost all of which include the SQL language. The main considerations for selecting a RDBMS are the following.

- What is the size of the implementation? – For small desktop databases, software application products such as Microsoft® Access® are suitable. For much larger client server-based requirements and web-based databases, software application products such as Oracle® Database, MySQL™ and Microsoft® SQL Server® are suitable.

Step by step: Using comparison and Boolean operators in SQL

1 These examples use the Rescue dogs table that was previously used to explain the basics of SQL (see Step by steps in the section SQL).

2 The comparison operators can only be used with numeric fields. So, for example, to find dogs born after 2010 you would enter:

```
SELECT * FROM Rescue_Dogs WHERE Year_of_Birth > 2010
```

You can of course use the other comparison operators to get different results.

3 The Boolean operators allow you combine selection criteria. The AND operator finds records which match both the criteria you set. Suppose you wanted a small male dog. The SQL statement would be:

```
SELECT * FROM Rescue_Dogs WHERE sex = 'M' AND approx_size = 'Small'
```

4 The OR operator selects records which match either of the criteria you set. So this statement will list all the male dogs and all the small ones (whatever sex they are):

```
SELECT * FROM Rescue_Dogs WHERE sex = 'M' OR approx_size = 'Small'
```

5 You can also combine criteria and operators, for example this statement will find the female dogs who are either black or white:

```
SELECT * FROM Rescue_Dogs WHERE sex = 'F' AND (colour = 'Black' or colour = 'White')
```

6 You can also use NOT in a WHERE condition. If you wanted to find all the dogs which are not the Doberman breed the command would be:

```
SELECT * FROM Rescue_Dogs WHERE NOT Breed = 'Doberman'
```

- Is proprietary or open source software preferable? – MySQL™ is open source, while products from Oracle® and Microsoft® are proprietary and require a licence fee to be paid.

Application design

As with any software development project, the database application 'front-end' needs to be designed, that is, the interface that users see, as well as the structure. Many database systems are accessed through a web-based front-end (user interface) so this is one area where this topic overlaps with website development.

Link

For more on principles that will be useful when designing the user interface for a database software application, see *Unit 6: Website development, Principles of website design.*

The user interface design will need to define the layout of each screen, including labels (text to guide the users), fields, controls (such as drop-down boxes and command buttons), error messages and colour schemes. There will also probably need to be software application designs that define the procedures which connect the user interface to the underlying database.

Database implementation techniques

Like any software development project there are a number of ways in which the database and associated applications can be designed and implemented. The approach taken with a particular project will depend on a number of factors including whether this is an entirely new database application or if it is replacing some existing computer or paper-based system.

Prototyping

The use of prototyping is one way to attempt to ensure that the eventual product (database solution) meets the users' requirements. Prototyping involves building a simplified version of the system with limited functionality. The prototype can then be demonstrated to the users to see if it matches their requirements. Prototypes can be further refined and added to and further demonstrations done for the users. However, the problem with prototyping is that it can delay the software development process as building, demonstrating and refining the prototypes can be time consuming. On the other hand, by not prototyping and not getting confirmation from the users that your design fulfils their requirements, you risk building a database solution that does not meet the users' requirements, therefore wasting time and money.

Data conversion

In cases where the new database system replaces an existing system, the issue of data conversion needs to be considered. Existing systems may contain very large amounts of data which may need to be reformatted and split into a different structure of tables and relationships.

Data may also need 'cleaning', that is, the removal of inaccurate, invalid or erroneous data that, over time, has been included amongst the valid data in the existing database.

These data conversion steps can be very time consuming if a large volume of data is involved.

Testing

Once the design and development work is complete, the database and its associated applications must be tested, using the test data and plans created during the design phase. If the system is large and developed in several parts, it may be possible to test each part as it is completed. However, once all the parts are complete, integration testing of the whole system must also be done.

Quality, effectiveness and appropriateness of the solution

The quality, effectiveness and appropriateness of the database solution design all need to be considered before it is implemented.

Quality

The quality of the design has a direct impact on the eventual quality of the completed database system. Although changes can be made to the design where faults or omissions are discovered during implementation, these can cause major delays to the project and cost additional money.

Effectiveness and appropriateness

The effectiveness and appropriateness of the solution is usually judged by the degree to which it meets the user requirements, which is why understanding those requirements is so important. The user requirements themselves may contain a number of differing views.

For example, the people who will use the system may have different requirements from those who are paying for the development of the database (they are not always the same people). These conflicts in user requirements should be addressed by the database designer early on through discussion with the people/organisation paying for the development of the database. They need to consider the needs of the users of the database, just as the designer does, because if they do not then users either will not purchase the database or not use it correctly.

Other factors

There are a number of other areas to be considered.

- Normalisation and relationships between data – The normalisation process should help to validate the design of the relationships between the data and the overall structure of the database.
- Correctness of data – It is very important that the data put into the database is as correct (accurate) as possible. Every opportunity should be taken to validate data. Also, the design of the user interface should, wherever possible, prevent the user from making invalid inputs. Clear error messages should be used, where appropriate, to make it clear why inputs are rejected.
- Data integrity – Careful consideration needs to be given as to how to maintain data integrity. This can include deciding whether or not to use the cascading update and delete features built into the database software to preserve referential integrity.

Reflect

Over the years, there have been a number of high profile database system developments which have failed for a variety of reasons. Perhaps the best known is the NHS patient records system – Google 'NHS system failure' for details. It may be that the user requirements were not well enough understood or the system was too complex.

How can you ensure, at the outset of a database development project, that it can be implemented successfully?

Design documentation

The design of a database needs to be documented so that the structure and component parts are implemented according to the design and to meet the user requirements. Also it is likely that, in the real world, the software development of the database system is designed and implemented by different people within a team.

In this section you will look, in detail, at the techniques and associated documentation that is created during the database design process.

Requirements of the brief

The first step in designing a database is to understand the client brief. This can be quite difficult as, often, the client will understand their business much better than the developer. The requirements need to be discussed with the client and a formal document, which describes the requirements in as much detail as possible, should be created and agreed on.

The requirements of the client brief are often the user requirements but they may also (or instead) be the requirements that the client has for the database, which may include constraints such as time for development and a budget.

The users of the database are known as the audience, and it is important to understand who they are and what they require from the database: that is, understand what their user requirements are. The purpose of the database means what it is for and what questions it needs to be able to answer – this should be part of the client brief.

Link

For more on the topic audience, purpose and client brief see *Unit 8: Computer games development, Audience, purpose and client requirements.*

Security and legal considerations

Many databases hold information about living individuals and, as such, come under the requirements of the Data Protection Act 1998 and The European Union (EU) Directive on Data Protection legislation.

When designing the database, the requirements in the Data Protection Act 1998 need to be considered.

This act stipulates that the data must be correct, it should be held securely and additional personal data not required for the application should not be collected

Correctness of data

Correctness of data is a major consideration for the database design and not just for the data protection requirements. Designers need to consider a number of different methods to try to ensure that the data that is recorded in the database is correct (accurate). This can include techniques such as validation of input data, as discussed earlier.

Securing data

Securing the database so that only those people who need to have access to the data can view or edit it can be done in a number of ways.

With a desktop database created with Access®, the database can be password protected and encrypted. This is particularly important if the database is stored on a laptop, which is vulnerable to theft.

Worked example: Dissford Arts Centre 1

A local arts centre has a small hall and a couple of studios which they rent out to music, drama and art groups to run workshops and performances. They have won a lottery grant to improve their facilities and build a small theatre and some music practice rooms. They want to replace the paper-based booking diary with a more sophisticated booking database.

The centre would like a prototype of the system to be developed before the complete database solution is implemented so that they can check that it meets their requirements.

The details of the requirements of the brief for the prototype database system are as follows.

- Audience – Initially, the arts centre staff will be the only users of the database, although eventually an online booking will be added.
- Purpose – The purpose of the database is to allow staff to search for and book free slots for the arts centre rooms. Initially, there will be just four rooms on the system: the large theatre hall, the small hall and two multi-purpose studios. Each room can be booked for an afternoon or an evening session.

The client's requirements are as follows.

- Users should be able to pull up a list of free slots (i.e. dates and sessions which are not already booked) for a given room within a date range.
- Users should be able to select and book a free slot.
- When booking a slot, users can select an existing client (i.e. person or organisation making the booking) or create a new client.

With a server-based database, a user access regime needs to be designed which restricts the access of users or groups of users to the database, so that they can carry out their jobs with regards to the database, but cannot do anything more to the database.

Link

For more on security and legal considerations, see *Unit 6: Website development, Legal and ethical considerations.*

For more about the Data Protection Act 1998, see the Information Commissioner's Office website, www.ico.org.uk.

Data structure designs

The first step in designing data structures will probably be to look at any paper-based records and to take these through the process of conceptual, logical and physical modelling described earlier. The end result of the process should be a detailed ERD showing how the tables are related and a data dictionary which defines the fields required for each table. Ensuring that the data which ends up in the database is correct is very important and there are a number of ways you can help prevent incorrect data being accepted. This includes the details contained in the data dictionary about field dataatypes, lengths and validation rules.

Data dictionaries

The data dictionary for a database needs to define a number of things about each table in that database.

- Field names – Each field within a table needs to have a meaningful name.

Link

See Naming conventions for more on how to choose meaningful names for fields.

- Datatypes – You need to select an appropriate datatype for each field. Datatypes help to validate data as they prevent accidental errors. For example, by setting a numeric datatype for a field, you cannot accidentally enter text or, if you set a date field, you cannot enter invalid dates. Access® provides a range of text and numeric datatypes plus special ones such as date, currency and Boolean (Yes/No).

Tip

Note that the short text datatype defaults to a length of 255 characters, which is quite large, so you might want to make it shorter.

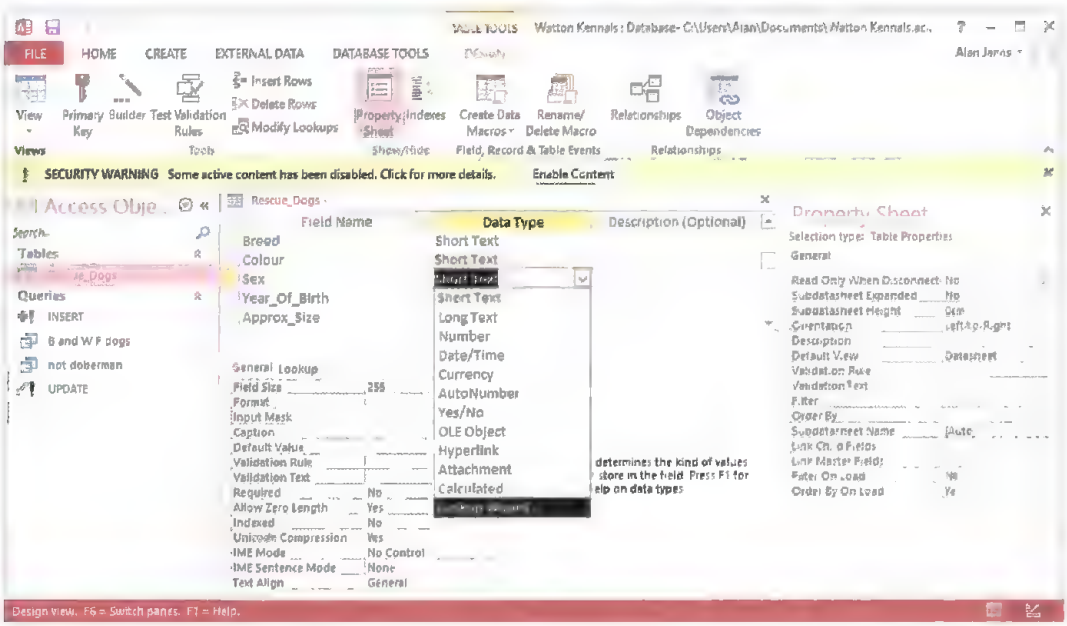
- Validation rules – You can also add a variety of rules to prevent invalid data being entered. For example, only 'M' or 'F' can be entered into a field recording a person's gender. This can use a short text datatype, of

one character length. The easiest way to make sure that only the values of 'M' or 'F' are entered is use a lookup table. Lookup tables work fine where there are a defined number of options for a field but would not work where there is a range of numeric or date values. In this case, a validation rule is required. The following Step by step will show you how to use lookup tables to restrict values entered into a field. The second Step by step will show you how to set validation rules.

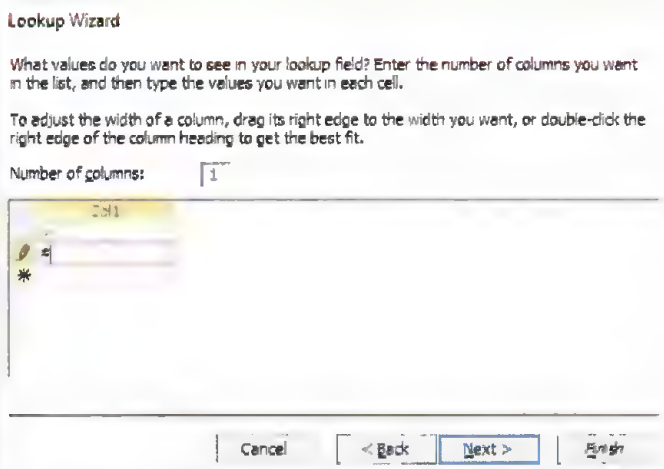
Lookup tables

Step by step: Lookup tables

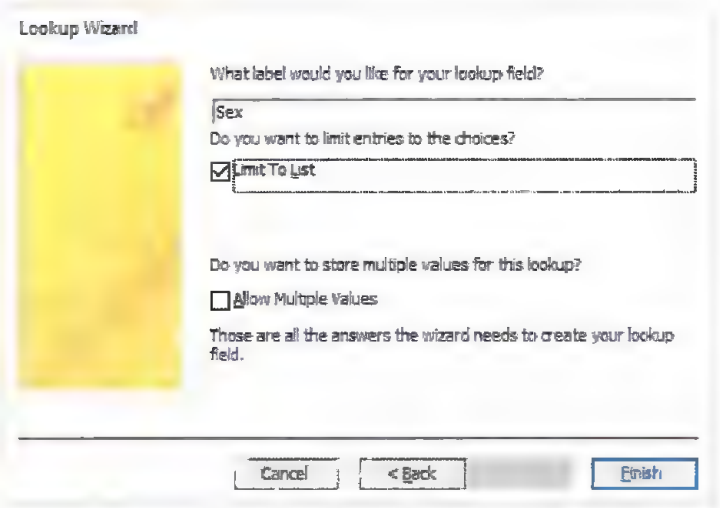
1 To use a lookup table to restrict the values that can be entered into field, first go to the Design View for the table and then for the field in question. Select Lookup wizard from the drop-down list of datatypes. In this example, we will use the Rescue dogs table that we worked with previously, as shown here.



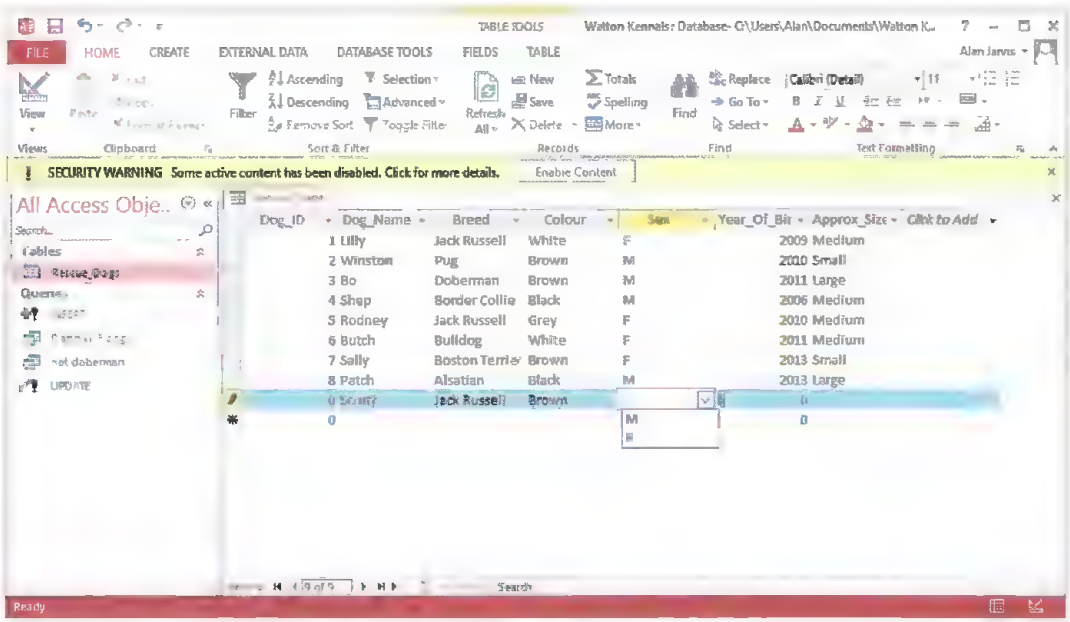
2 The Lookup wizard will then ask you if you want to look up the values in another table or query (which can be useful if there are a lot of values) or if you want to type them in yourself. As we only require 'M' or 'F', typing the values in yourself is the easiest option, so select this one and click Next. You will then see this next step of the wizard, as shown here.



3 On this screen, in the Lookup wizard, you type the required values and then click Next. The final stage of the Lookup wizard is shown here. Enter a label for the field and, in this case, tick the box which asks if you want a Limit to the list, because you do not want any values other than 'M' or 'F' in the field.



4 Once the Lookup wizard is complete, you need to save the table. Then, if you return to Datasheet View and enter a new record, the Sex field is no longer a free text entry field but a drop-down list with the values you entered, as shown here.



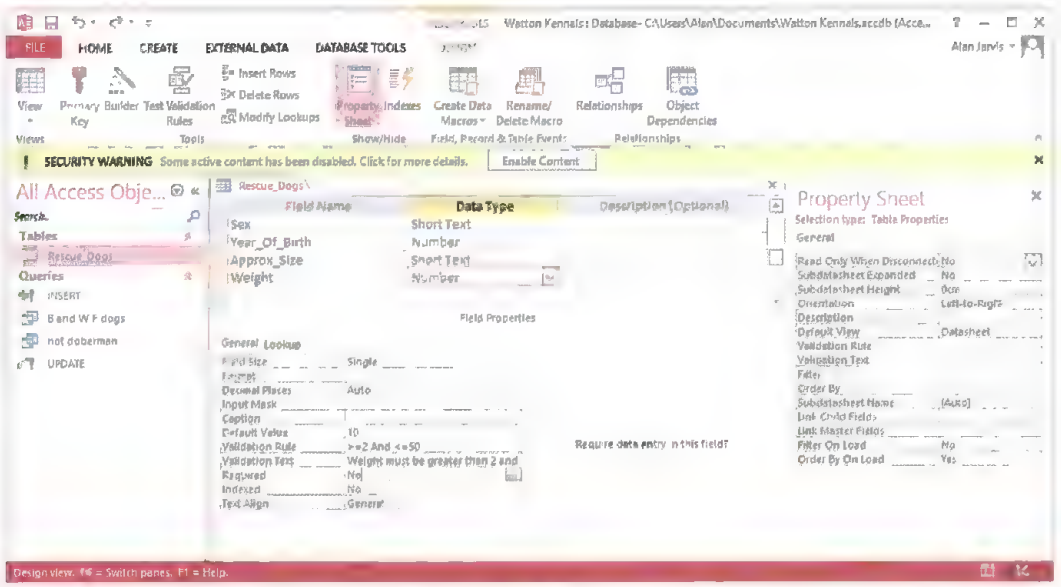
The same technique (as that shown in the Step by step here) could be used on the Rescue dogs table for the Approx_size field to limit the options. Remember that if, for example, a dog's details were entered and its size is misspelt, for example 'Medum' rather than 'Medium', then that dog would not appear in any query searches for medium dogs.

Validation rules

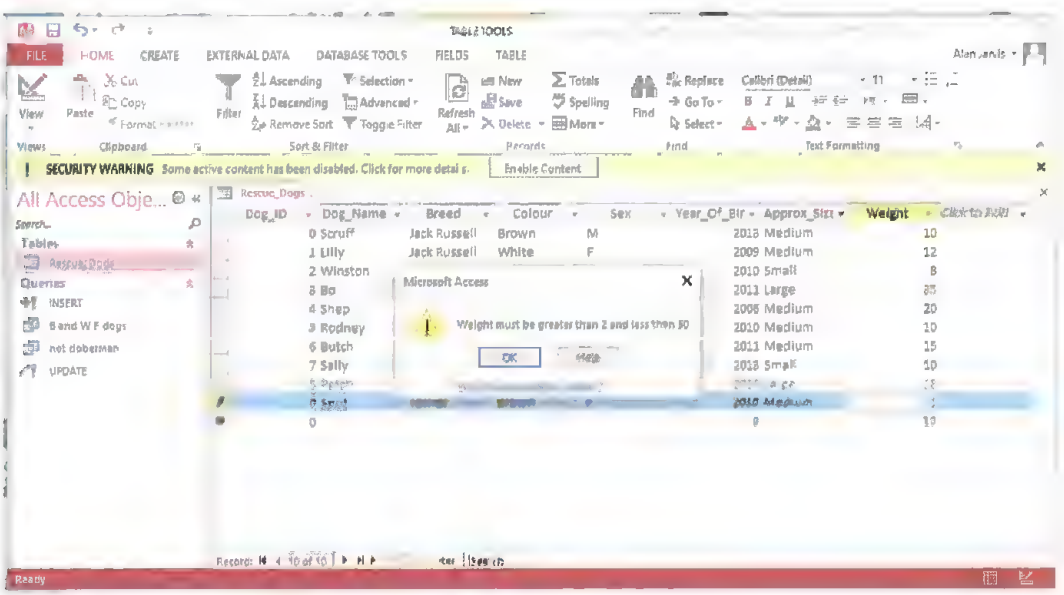
Lookup tables work fine where there are a defined number of options for a field but would not work where there is a range of numeric or date values. In that case, a validation rule is required.

Step by step: Validation rules

Again, using the example of the Rescue dogs table most recently used in the Lookup tables Step by step, suppose that you had a field for the dog's weight in kg. Dogs come in a wide variety of sizes but are rarely less than 2 kg or more than 50 kg. This screen shows how you would create a validation rule to restrict values to greater than 2 and less than 50.



Note that the default value for this field has also been changed from 0 to 10, so it does not violate the validation rule. The default value is the value that appears in the field before the user types anything. The validation text is the error message that appears if the rule is violated, as shown here.



Care must be taken creating validation rules as, unless the validation text is clear, users may not understand why they cannot enter certain values in a field. Also, note that if you change the validation rules on an existing database table which contains data, all the records in the table need to be checked to ensure they do not violate the rule, which can take a long time if there are a lot of records. This is one of the reasons why it is important to get the design of the database right before you put lots of data in it.

Worked example: Dissford Arts Centre 2

Conceptual model

In order to create the design for the Dissford Arts Centre booking system database, the first step is to work on the conceptual model. From the current paper-based diary shown earlier, we can extract the following fields.

- Date (the date of the booking).
- Room name (can be Large theatre, Small hall, Studio 1 or Studio 2).
- Session (can be afternoon or evening).
- Client name.
- Client telephone number (plus other details such as address etc.).

If we put this into a single table and add some actual data, it would look something like this.

Date	Room	Session	Client name	Client tel	Client address	Client town
9/11/15	Large theatre	Afternoon				
9/11/15	Large theatre	Evening	Drama Club	012345678	10 High St	London
9/11/15	Small hall	Afternoon				
9/11/15	Small hall	Evening	J Smith	079999999	28 Station Rd	Norwich
9/11/15	Studio 1	Afternoon	Art Society	012398765	66 Main Rd	Watton
9/11/15	Studio 1	Evening	Art Society	012398765	66 Main Rd	Watton
9/11/15	Studio 2	Afternoon				
9/11/15	Studio 2	Evening	Rock band	079988888	3 London Rd	Thetford

This shows all the booking slots for one day (9 November). Note that some of the slots are free (no client details recorded).

Based on this information, it is fairly clear that there are three entities: Clients, Rooms and Booking slots. Clients 'reserve' booking slots and Rooms 'are booked'. The conceptual ERD looks like this (see Figure 2.7).

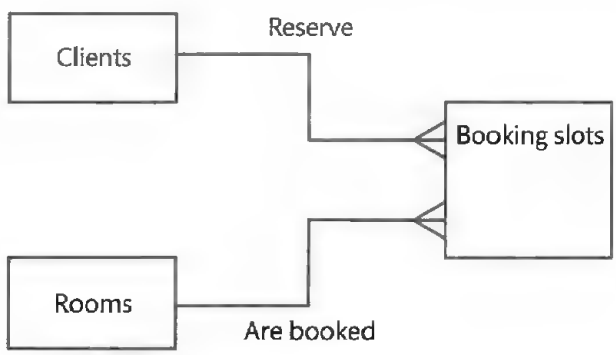


Figure 2.7: Conceptual ERD

Logical model

From this conceptual model of the database, we can now develop the logical model.

We know that each table needs to have a primary key and that this key needs to be unique and that it is best if it takes a numerical value which does not have meaning outside the database (e.g. Client name does not make a good primary key). Therefore, our data table should now look like this (for simplicity the client address and town have been left out).

Booking slot ID	Date	Room ID	Room name	Session	Client ID	Client name	Client tel
1	9/11/15	1	Large theatre	Afternoon			
2	9/11/15	1	Large theatre	Evening	1	Drama Club	012345678
3	9/11/15	2	Small Hall	Afternoon			
4	9/11/15	2	Small Hall	Evening	2	J Smith	079999999
5	9/11/15	3	Studio1	Afternoon	3	Art Society	012398765
6	9/11/15	3	Studio1	Evening	3	Art Society	012398765
7	9/11/15	4	Studio2	Afternoon			
8	9/11/15	4	Studio2	Evening	4	Rock band	079988888

From this we can create our logical ERD (see Figure 2.8).

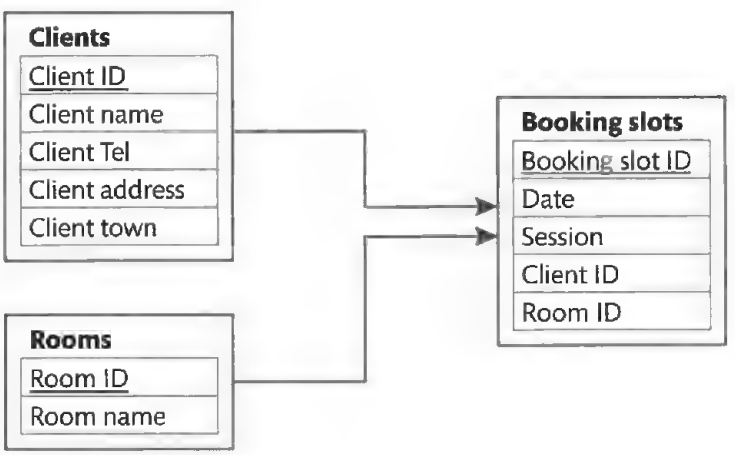


Figure 2.8: Logical ERD

Data dictionary

We can also create the data dictionary for each table, which describes each table and their fields in much more detail.

Data dictionary: Clients table

Field name	Key?	Datatype	Validation
Client_ID	Primary	Integer	Automatically created by Access
Client_name		Text, 25 characters	
Client_tel		Text, 12 characters	
Client_address		Text, 25 characters	
Client_town		Text, 25 characters	

Data dictionary: Booking slots table

Field name	Key?	Datatype	Validation
Booking_slot_ID	Primary	Integer	Automatically created by Access
Date		Short date	
Session		Text, 10 characters	Can only be afternoon or evening
Client_ID	Foreign key from Clients table	Integer	Must exist on Client table
Room_ID	Foreign key from Rooms table	Integer	Must exist on Rooms table

Data dictionary: Rooms table

Field name	Key?	Datatype	Validation
Room_ID	Primary	Integer	Automatically created by Access
Room_name		Text, 25 characters	

For the sake of simplicity, only a small number of fields have been included in these tables; in a real database there are likely to be many more fields.

Naming conventions

There are a range of conventions (rules) which can be used for naming database objects such as tables, fields, queries etc. The most important rules are the following.

- Do not use spaces – rather than First Name, use First_Name or FirstName. Access® allows you to use spaces, but it can cause problems if you use the database on a web server or with other database software.
- Do not use names that are reserved words – such as 'date'. (Access® will not allow you to do this anyway.)

Entity-relationship designs

ERDs need to be created at the conceptual design phase and refined in the logical design phase. Examples of ERDs are given in the worked example above.

Normalisation

As described earlier, the process of normalisation is used mainly as a validation technique to check if the conceptual and logical model of the database design is correct. The normalisation process needs to be documented with descriptions of the way in which each stage was completed.

Worked example: Dissford Arts Centre 3

The normalisation for this data is fairly simple. If we look at the first table under the heading 'Logical model', in the previous Worked example, then the Booking slot ID is a suitable primary key, and since it is not a compound key this data is already in 2NF.

Remember that the requirement of 3NF is that if non-key attributes depend on other attributes within the data, then they need to be moved to another table. In the data shown in this table, if we know the room ID then we can identify the room name, so these two need to be moved to a separate table.

The same is true of the Client ID – if we know this, we can uniquely identify the client name and their telephone number. Therefore the normalisation exercise produces the same result as shown in the logical ERD, validating that our design is correct.

User interface design

The user interface can usually be broadly divided into the parts of the interface used for data entry and those used to present information to the user.

Data entry

The most important aspect of designing a data entry interface is to ensure that, as far as possible, the data that is entered is correct. There are a number of ways to ensure the correctness of data.

- Validation – Various validation methods can be used. For example, dates can be validated to ensure that they fall within the correct ranges (so a date of 31st February is clearly recognised by the database as being incorrect). The use of a 'date picker' (miniature pop-up calendar) is one way to make date entry easy and to help reduce errors.
- Verification – Verification is used where the correctness of data entered is essential. It involves entering the data twice and checking that both entries are the same. It is often used for setting passwords and entering email addresses into forms.
- Calculated fields – Calculated fields can also be used to check numerical data. 'Check digits' are a commonly used example of a calculated field that are used by banking applications to avoid errors in the manual inputting of bank account and debit/credit card numbers. This involves the addition of one or more digits to a number which are calculated from the preceding numbers.
- Input masks – Input masks can be used where the data follows a pre-set format, for example NI numbers. These always consist of two upper case characters, followed by six numbers and a final single upper case character. The mask ensures that the data that is entered into the NI field must fit this pattern.
- Directed input – Directed input can be implemented where only a fairly small range of different values is acceptable. An example would be a value for gender, which can only be 'Male' or 'Female'. Methods of directed input include radio buttons (where there are only very few values) and drop-down lists.

Task automation

The user interface design should make the user's life easy and should automate procedures whenever possible. Fortunately, Access® provides a number of facilities that make automation of many tasks quite easy. However, details of how each task will be automated should be included in the design. Some of the areas where automation could be used include the following.

- Imports – Data may need to be imported into a new database from an existing one, so a design needs to be produced that maps the tables and fields across from the old to the new database. Access® provides a step-by-step import facility, which is covered in the Step by step: Importing data and using an action query to manipulate it. You can save the import steps so that you can run the automated import any time you need to.
- Updates – Automated updates to a table can be done using action queries. These are a special type of query which as well as selecting data based on certain criteria can also modify data. Action queries used to update data can be designed in the same way as select queries. An example of this is the Worked example: Dissford Arts Centre 2.
- Deletions – As with updates, action queries can also be used to automate the deletion of unwanted fields or records.

Data reports and presentation

The most important design consideration for the parts of the user interface that present data to the user (such as queries and forms) is that the data is reported/presented in a clear way with the correct fields shown in the correct order.

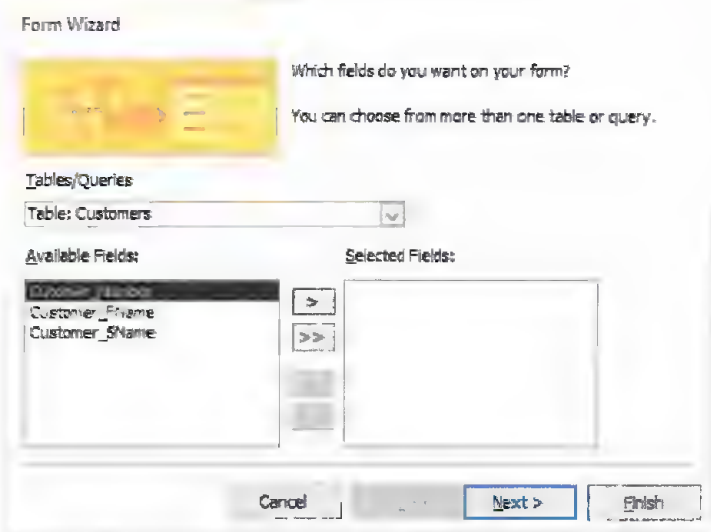
The use of colours, lines and boxes within the tables may also help the user to understand the data. As well as presenting information on the screen, many databases will need to produce printed reports.

Step by Step: Creating forms

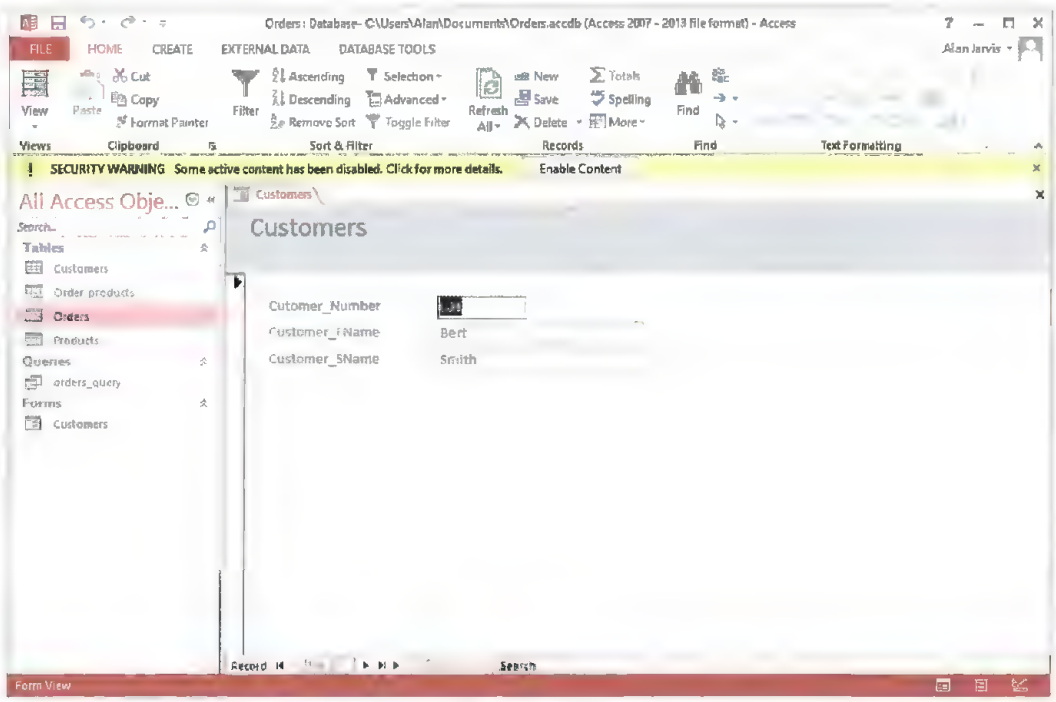
3 Steps

1 Access® provides a simple way for you to create forms to use for data entry or displaying data. This example uses the online orders database that was discussed earlier.

2 To create a form for the Customers table, first make sure that you have the Online orders database open and the Customers table selected. Then go to the Create menu and choose the Form Wizard icon. This will take you through a step-by-step process to create a simple form. First, it will ask you which fields you want to be included in the form, as shown here.



3 In this case, you want all the fields, so click the double arrow to move them all over to the selected fields box and then click Next. Then it will ask you what layout you want. This is something you can experiment with to see what kind of results you get, but for now just leave 'Columnar' selected and click Next. Finally, it will ask you what text you want for the title of the form. The default setting of the table name (Customers) is fine. Also leave the default setting selected which will open the form and view data (rather than opening it in design view). Click the Finish button and you should see the form displayed, as shown here.



Worked example: Dissford Arts Centre 4

One of the main requirements of the prototype database system is that users can enter a date range and then see a list of all the available slots for rooms during that period, and then proceed to book one of the free slots. The user interface design for this process could look something like this.

Step 1: The user enters the date range for when they want to see the available slots, as shown in Figure 2.9.

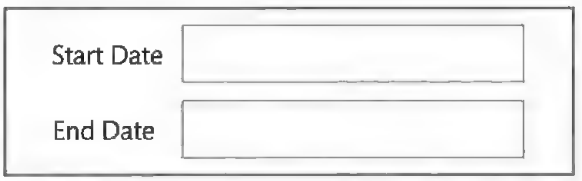


Figure 2.9: Date range data entry design

Step 2: The user is presented with a list of available booking slots as shown in Figure 2.10.

Available slots

Booking slot ID	Slot Date	Session	Room name	
nnnnn	nn/nn/nn	Xxxxxx	xxxxxxx	Book
nnnnn	nn/nn/nn	Xxxxxx	xxxxxxx	Book
nnnnn	nn/nn/nn	Xxxxxx	xxxxxxx	
nnnnn	nn/nn/nn	Xxxxxx	xxxxxxx	

Figure 2.10: Available booking slots design

Step 3: The user clicks a button next to the available slot they want to book and is presented with a form on which they complete the booking, as shown in Figure 2.11.

Make Booking

Booking slot ID	nnnnn
Slot Date	nn/nn/nn
Session	Xxxxxx
Client ID	nn
Room ID	nn
Room name	xxxxxxx

Figure 2.11: Make booking design

Extracting and presenting data

One of the main benefits of using a relational database is its ability to extract and present data in many different ways.

Queries

Queries provide the main way to extract data from a database. We have looked briefly at how to use SQL to extract records from a table which match certain criteria, but there are many more powerful features you can make use of.

The visual interface that Access® provides for creating queries can be easier to use than SQL and it also provides a number of useful features. You can, of course, always look at the SQL that is created by selecting the SQL View.

The same Online orders database that was used for the normalisation exercise will be used as an example. This database has four tables: Orders, Customers, Products and Order-products. Given the way that the database is structured, the details of an individual order are not stored in one table, but spread across the tables. A query can therefore be used to bring all the information about an order together.

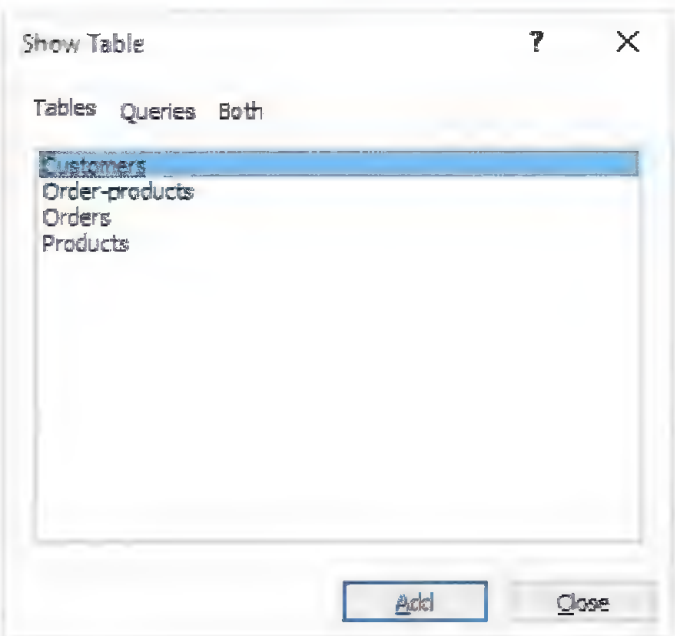
Link

See The stages of normalisation.

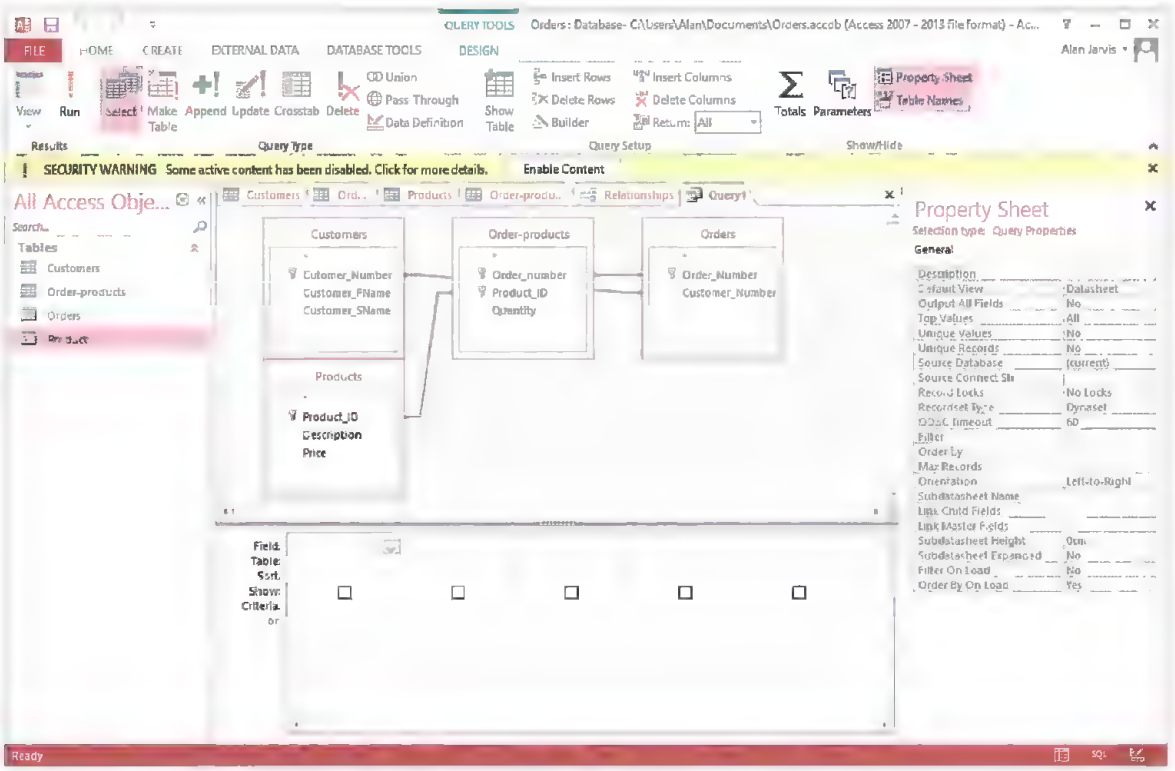
Step by step: Queries

6 Steps

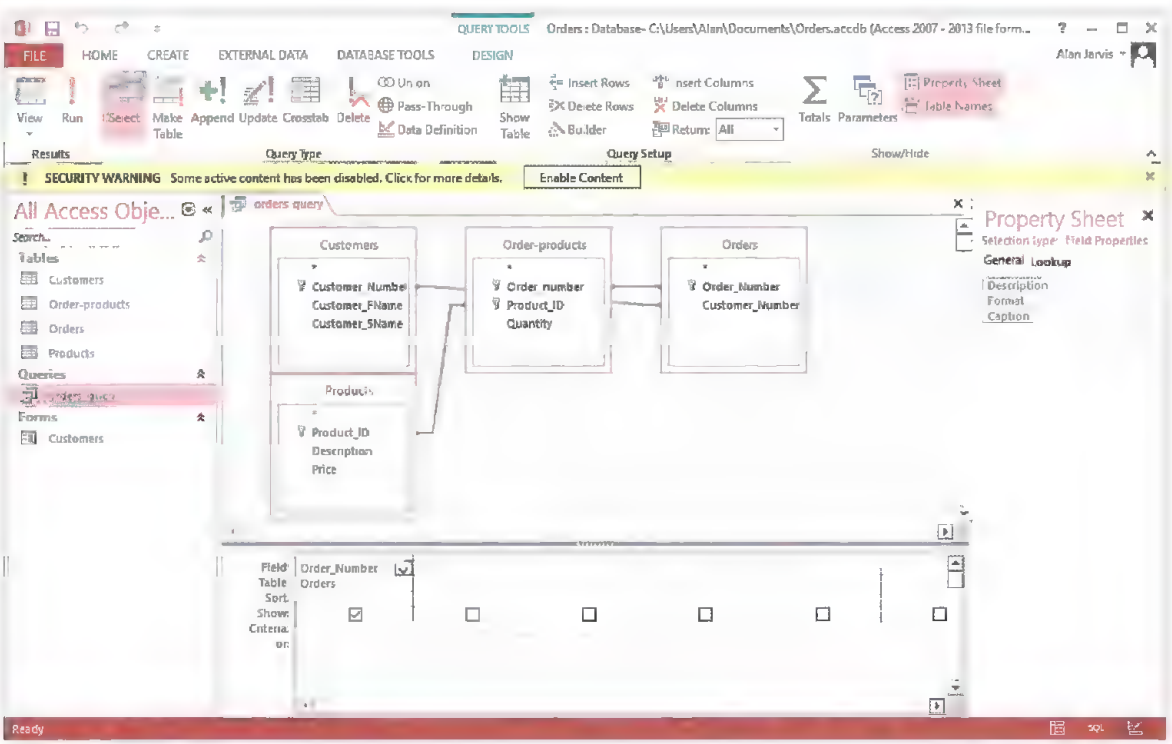
1 To create a new query in Access®, first, go to the Create menu and choose the Query Design icon in the toolbar. Access® will then display the Show Table Dialog shown here.



2 Click on each of the tables and then click Add. Then click close and you will see the Query Design View, as shown here.



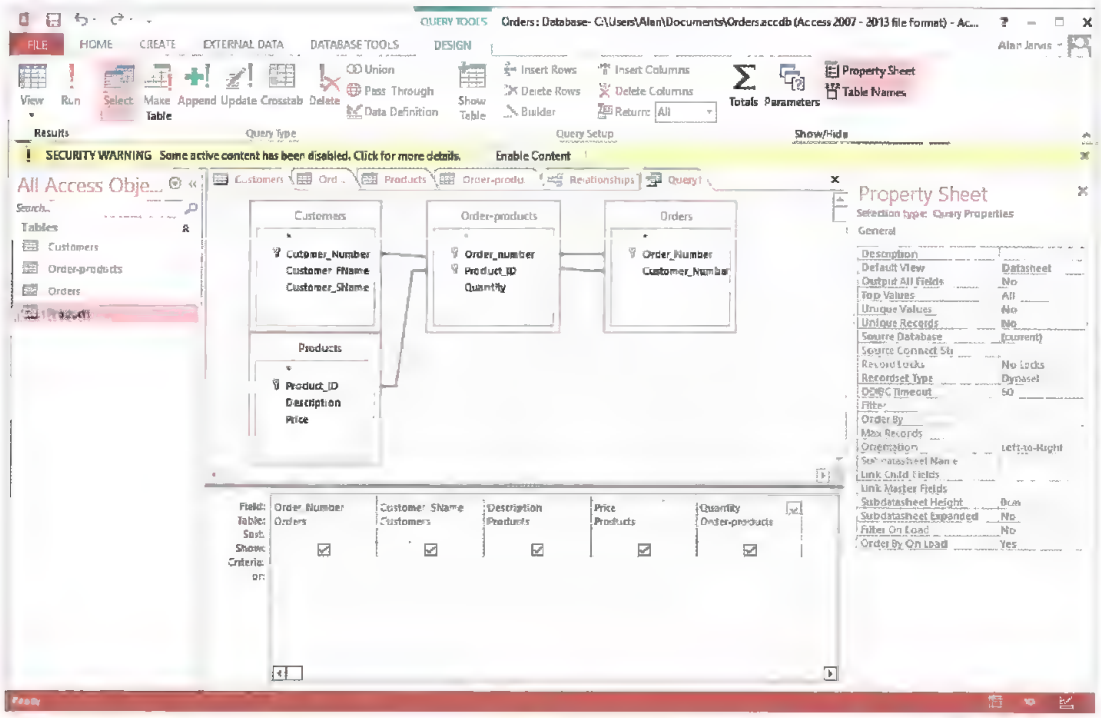
3 From the Orders table (in the top part of the Design View), click and drag the field 'Order_Number' down into the first 'Field' column of the query grid in the lower part of the display, as shown here.



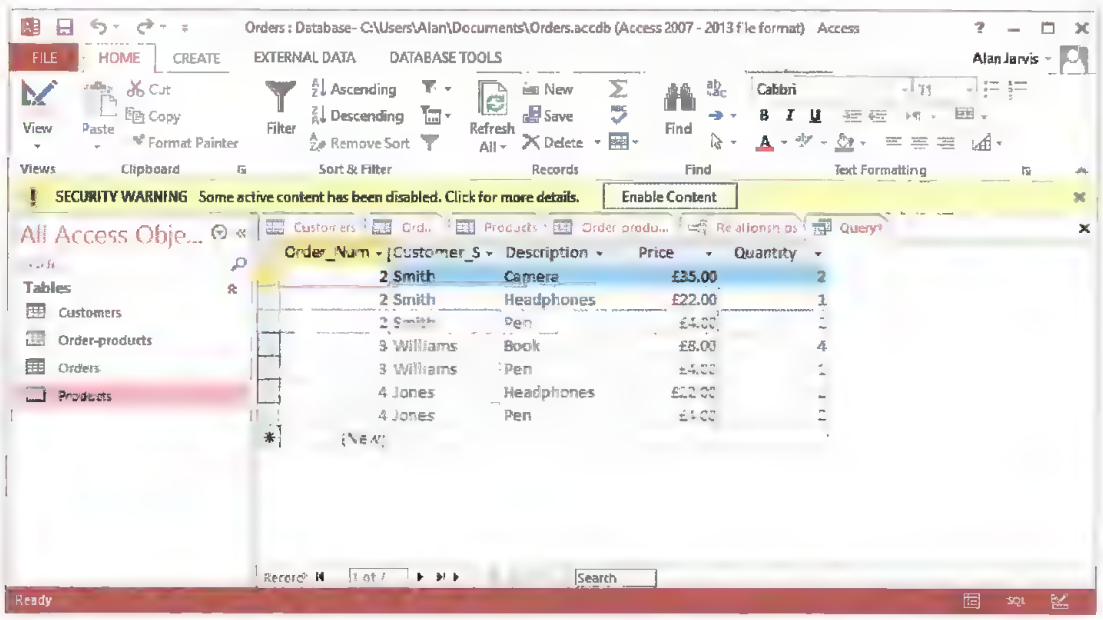
4 Next do the following:

- from the Customers table, drag the field Customer_SName into the next column of the query grid
- from the Products table, drag the Description and Price fields into the query grid
- from the Order-products table, drag Quantity.

The query grid should now look like this.



5 To run the query, click the Run icon in the toolbar and you should see the results, as shown here.



6 Save the query by right clicking in its tab and choosing Save. Give the query a meaningful name such as 'Orders_query'.

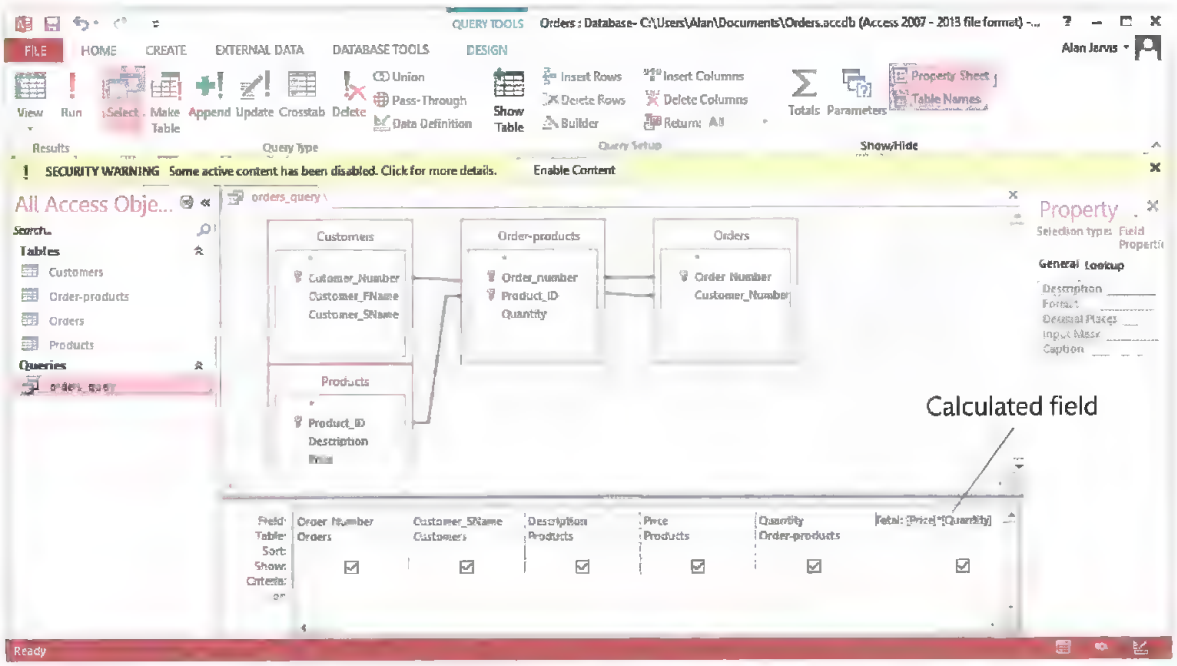
Calculated queries

You can also create calculated fields in queries. For example, in the query just created it might be useful to multiply each product price by the quantity to get the total cost of each product ordered.

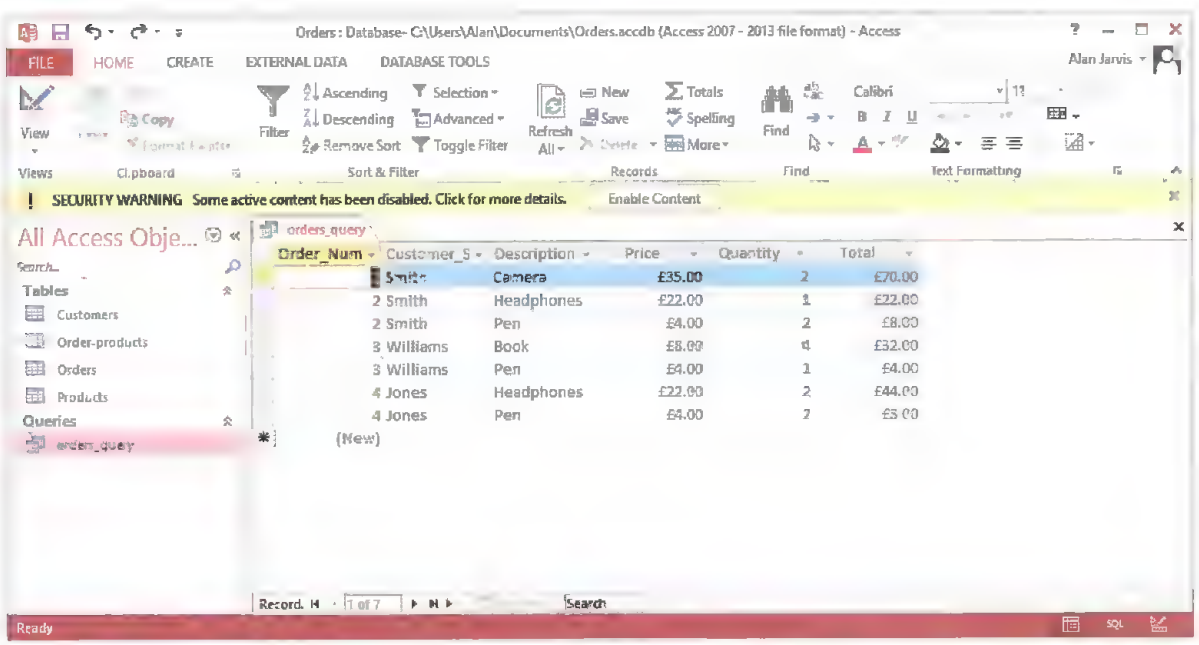
Step by step: Calculated fields in queries

To do this, open the previous Step by step, 'Orders_query', and return to the Design View. In the empty right-hand column of the query grid, type the following expression:

Total: [Price]*[Quantity]



If you now run the query, the added field shows the total for each item ordered, as shown here.

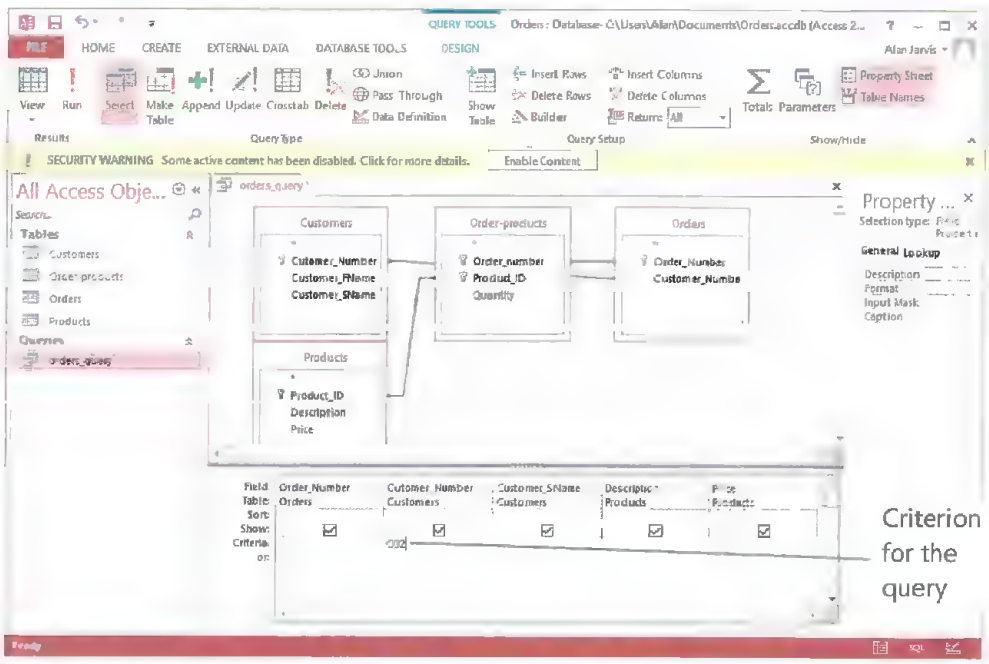


Queries using multiple criteria

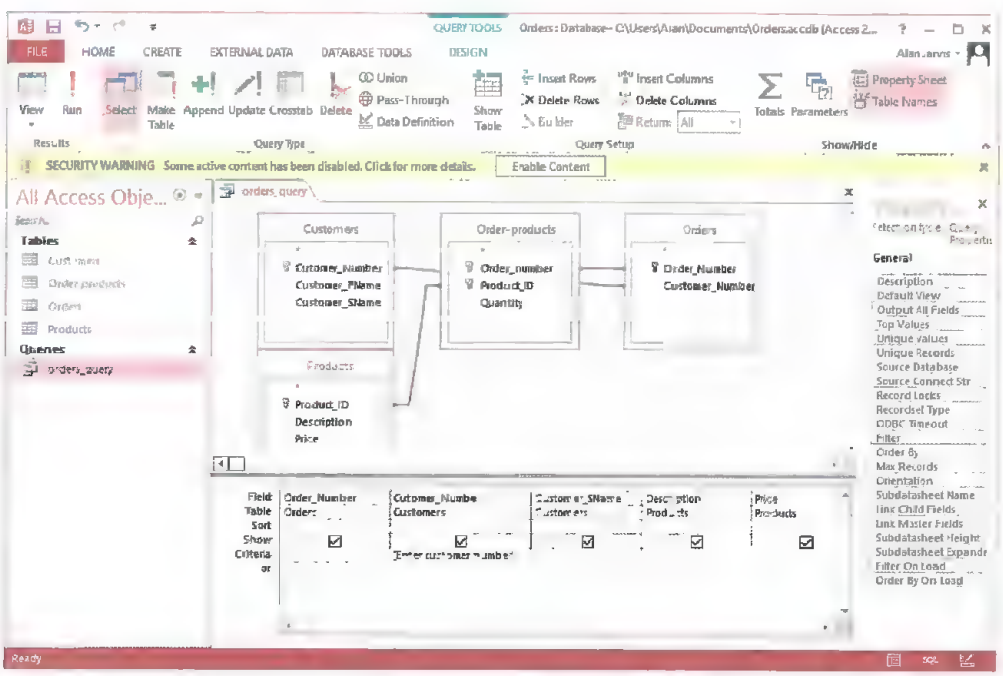
As well as typing the criteria that you wish to apply to a query directly into the query grid, a query can also pick up values used as selection criteria elsewhere. The simplest method is to use a pop-up dialog box to collect values for the criteria from the user.

Step by step: Queries using multiple criteria

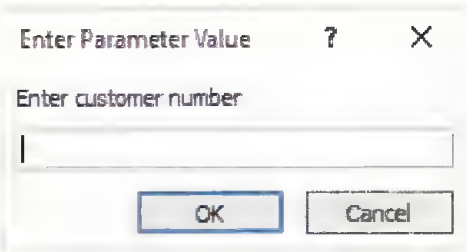
1 For example, suppose you wanted to use this query (from the previous Step by step) to look up a particular customer's orders. You could just add the customer number from the Customers table to the query grid and then type the customer number you want selected in the Criteria column of the query grid, as shown here.



2 This example would just display the orders for customer number C02. You can get the query to display a dialog box so that you can type in a different customer number each time the query runs. To do this, enter the prompt you want in the dialog box, surrounded by square brackets, as shown here. This is known as a parameter query.



3 Now when you run the query, you will see the dialog box, as shown here.



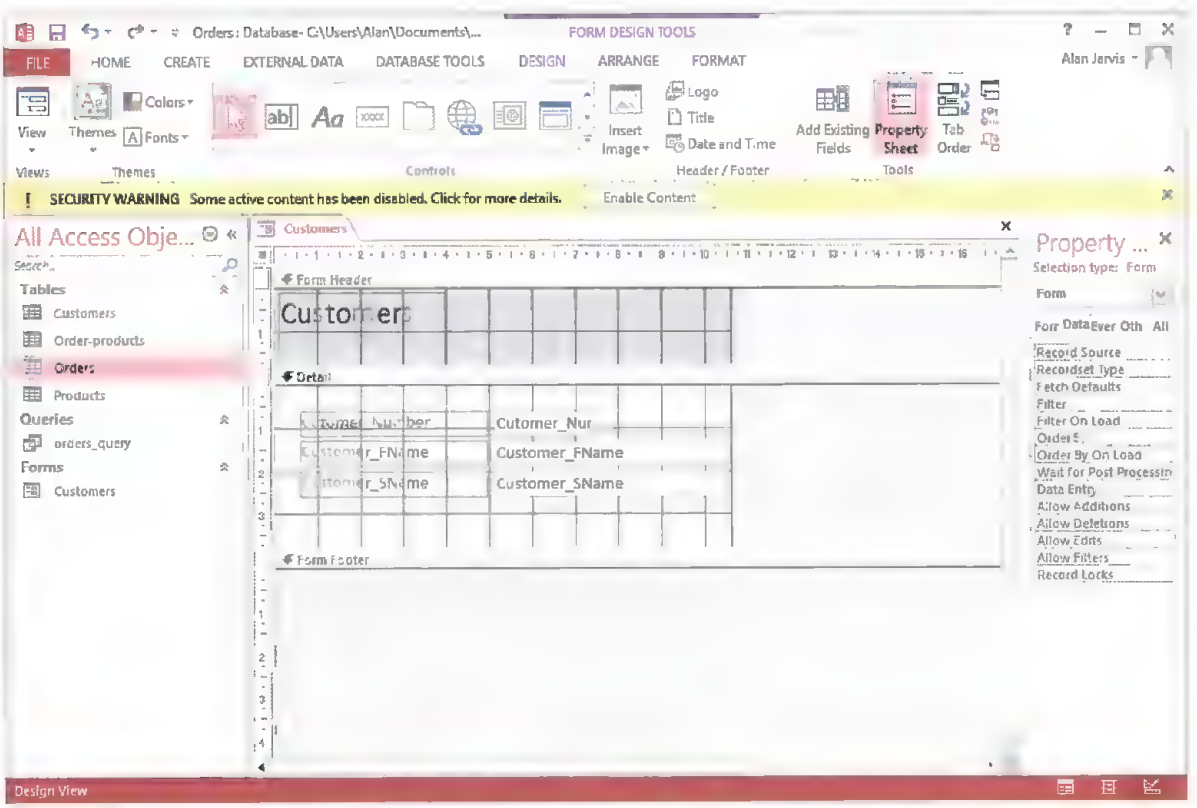
4 When you enter a valid customer number, it will display only matching records.

Query form values

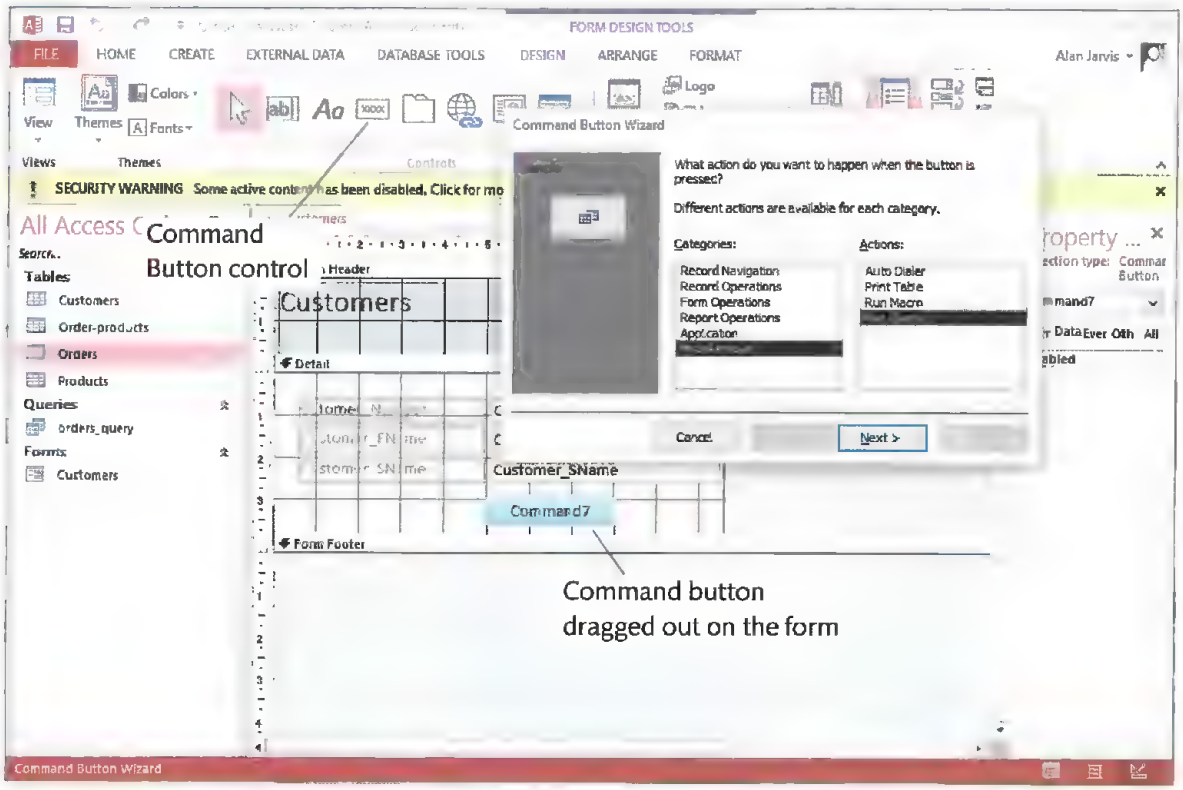
It would be even more useful, in terms of creating an easy-to-use application, if the query could pick up a value from a form.

Step by step: Query picking up a value from a form

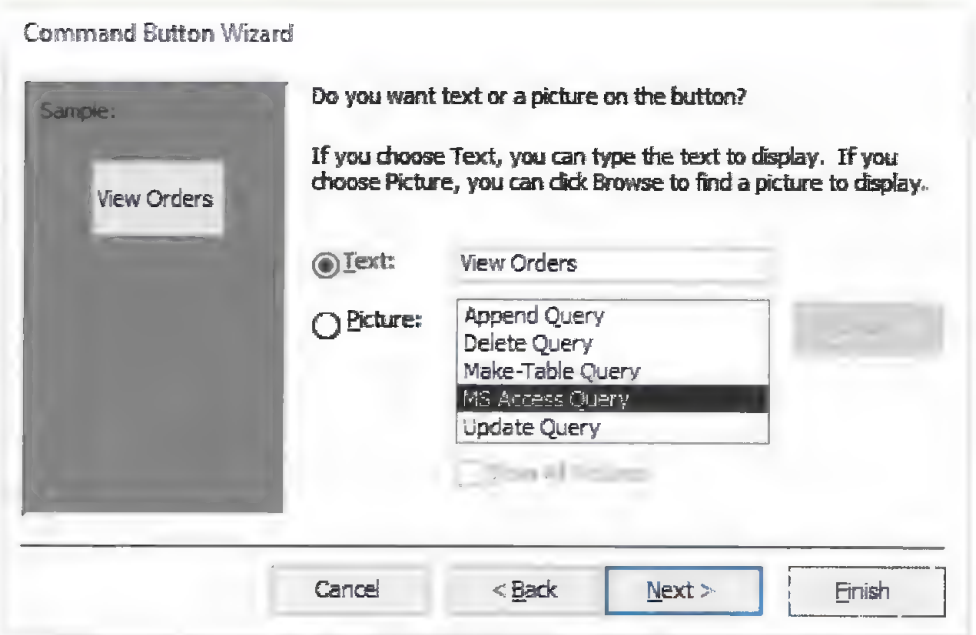
1 In the Step by step: Creating forms, we created a simple form for the Customers table. Now we are going to add a button which will open the query showing the orders that a customer has placed. To add the button, open the form that you created earlier, then click the View icon and choose Design View. This will display the form in Design view, as shown below. Drag the Form footer down a little to create some space for the button.



2 Now, in the Controls section of the toolbar, click the Command Button control and drag out a button on the lower part of the form. This will open the Command Button Wizard, as shown here.



3 Choose Miscellaneous in the list on the left-hand side and then Run Query on the right-hand side. Click Next and choose the name of the query you are working on. Click Next again to select Text to appear in the box. Enter something like 'View Orders', as shown here.

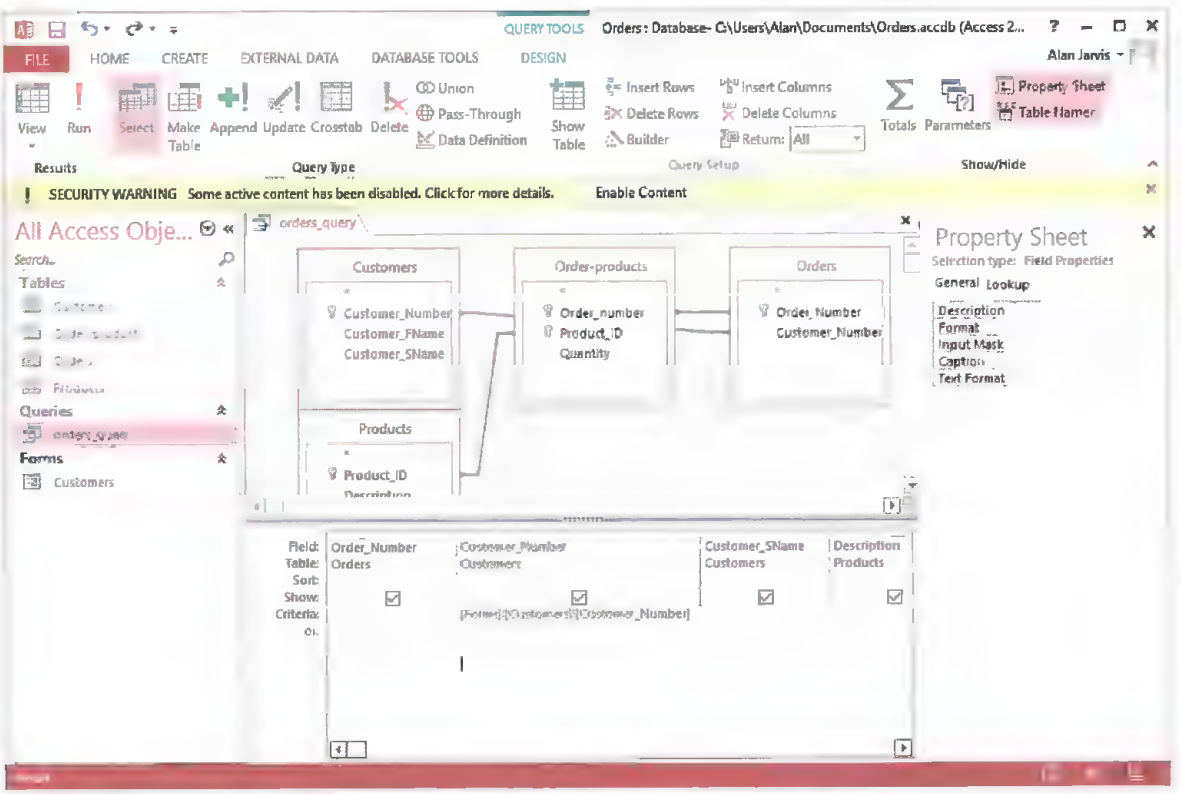


Now you can click Finish and the button is added to the form. At the moment, the button will show the query, but will show all the orders for every customer. We want to transfer the current Customer_Number field to the query criteria. Save the form and open the query in Design View. The expression we need to add in the Criteria column is as follows:

[Forms]![Customers]![Customer_Number]

This tells the query to obtain the criteria from a form called 'Customers' and a field called 'Customer_Number', as shown here.

Note that the query now will not work on its own; it will only work when opened from the form.



Save and close the query. Open the form again, use the navigation buttons at the bottom to select a particular customer, then click the button and the query will display only those orders for the current customer displayed on the form.

Worked example: Dissford Arts Centre 5

You have already completed the user interface design for the main booking process (finding an available booking slot and then making a booking in a free slot). Now you will need to use a query to find free booking slots. You can use a parameter query to select the date range of the booking slots and, by setting a criteria of only those booking slots which have no client ID in them (i.e. they have not already been booked), the query will select only available booking slots. Since Client_ID is a numeric field, it has a zero in it if no booking has been made. We can outline the design for the query like this (another possibility would be to look at the SQL for the query by selecting SQL view).

Query name:	Find_slot				
Tables:	Booking_slots	Rooms			
Query grid:					
Field	Booking_slot_ID	Date	Session	Room_name	Client_ID
Table	Booking_slots	Booking_slots	Booking_slots	Rooms	Booking_slots
Sort		In date order (ascending)			
Criteria		Between start and end dates			0
Notes					(does not need to be displayed)

Action queries

All the queries mentioned so far have just been used to select and display data, but Access® also supports action queries, which can make changes to data. Being able to make changes to data is useful, for example if you needed to update every record in a large database.

There are various types of action query:

- an update query can update every record in a table that matches some criteria
- a delete query can delete records which match certain criteria
- an append query can append (add) records from one table to another
- a make table query can create a new table based on records from an existing table.

Action queries can be useful if you need to create a set of related tables from a raw set of data held in a single file.

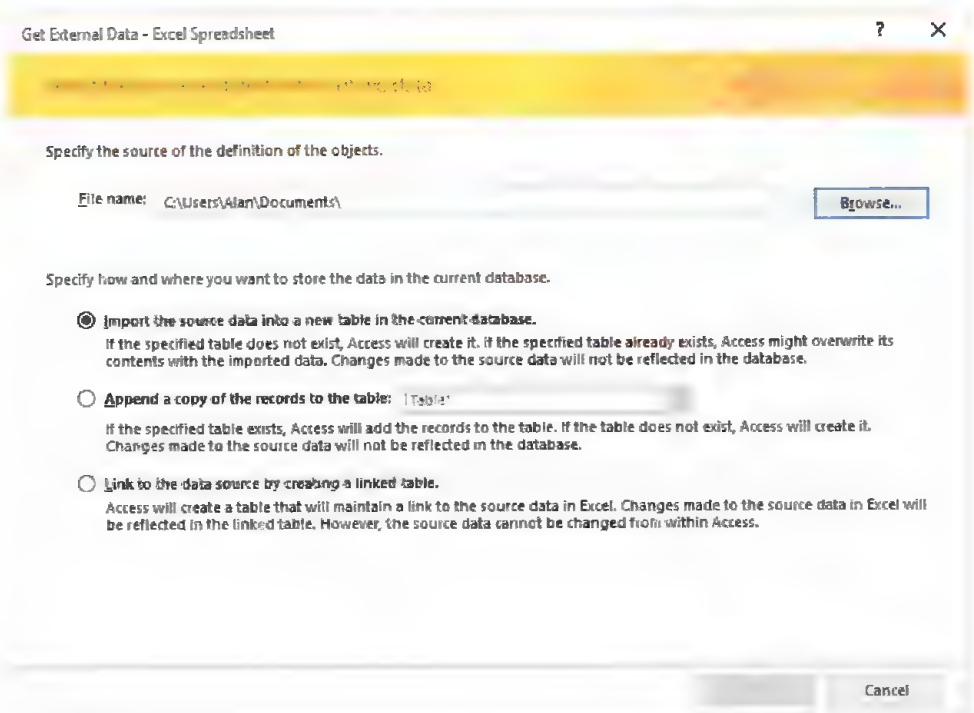
Step by step: Importing data and using an action query to manipulate it

1 It is likely that, in the assessment for this unit, you will be given a file of raw data which you need to import and split into tables, following the design you create for the database. Look at how this can be done with the 1NF version of the data that was used in the normalisation example earlier (see The stages of normalisation).

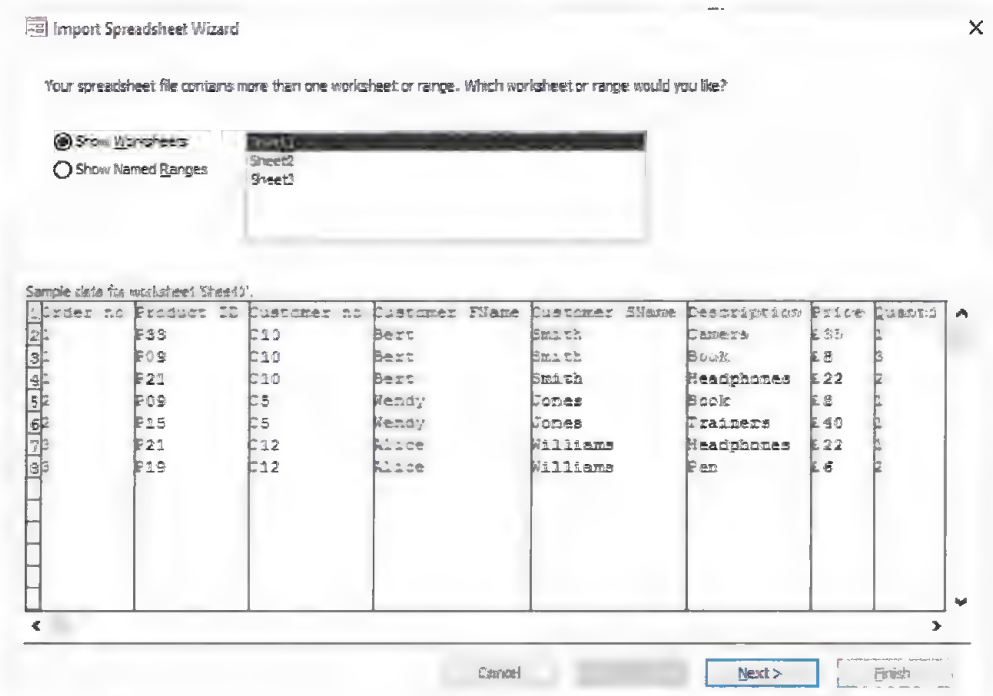
Here is a reminder of what the data looks like.

Order no	Product ID	Customer no	Customer FName	Customer SName	Description	Price	Quantity
1	P33	C10	Bert	Smith	Camera	£35	1
1	P09	C10	Bert	Smith	Book	£8	3
1	P21	C10	Bert	Smith	Headphones	£22	2
2	P09	C5	Wendy	Jones	Book	£8	1
2	P15	C5	Wendy	Jones	Trainers	£40	1
3	P21	C12	Alice	Williams	Headphones	£22	1
3	P19	C12	Alice	Williams	Pen	£6	2

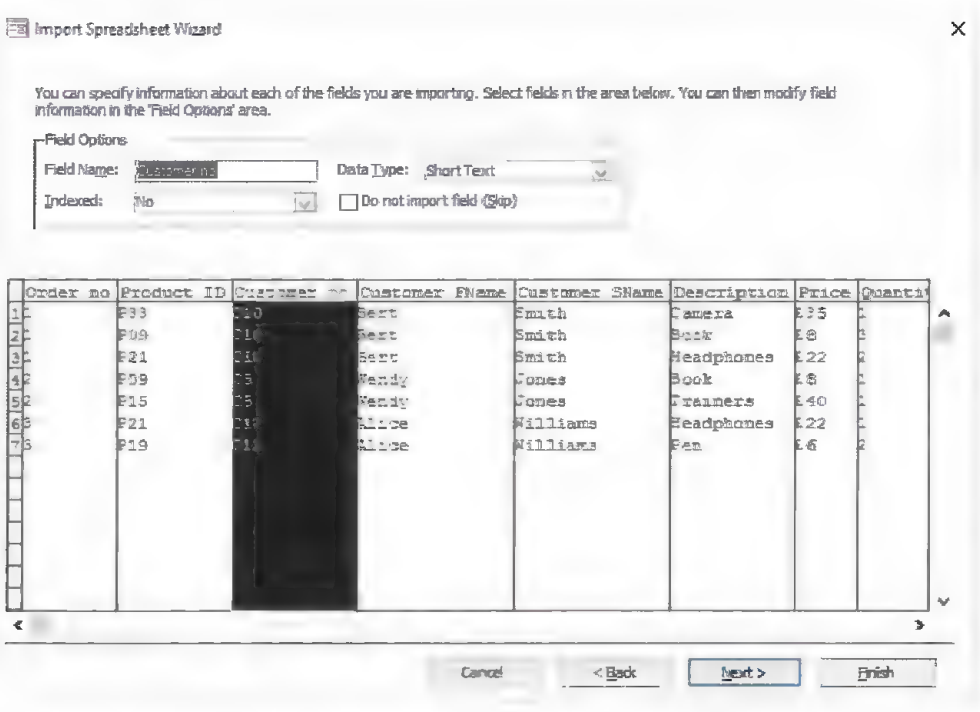
2 You can import various types of file into Access® including text files and Microsoft® Excel® spreadsheet files. First, open a database file (or create a new one) into which you want to import the data. Then, from the External Data menu, click on the icon for the type of file you want to import, which is Excel in this example. First, you will see the dialog box, as shown here.



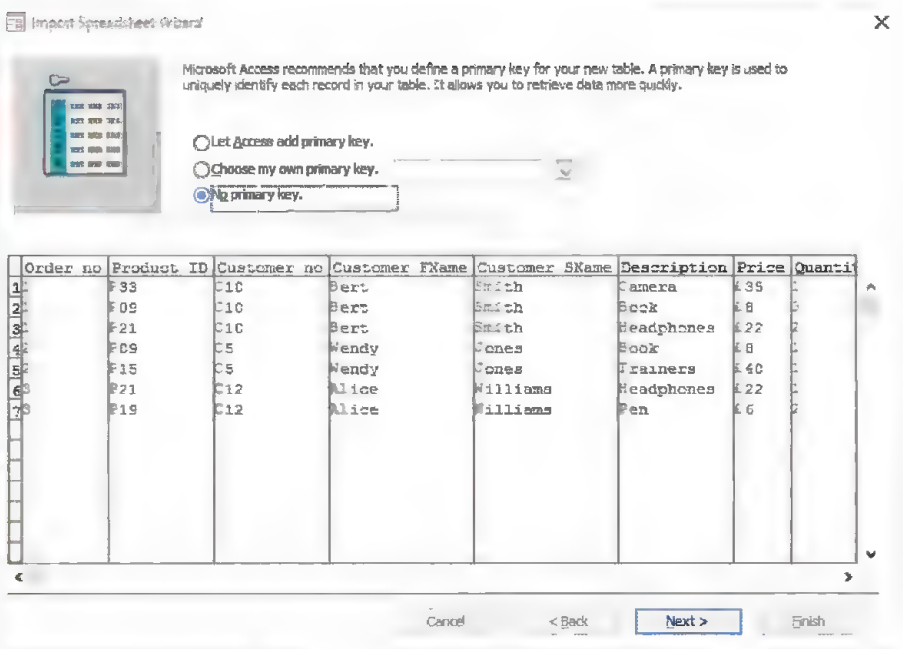
3 Browse to find the Excel® file you want to import and then click OK. Next you will see a preview of the data from the chosen file and you can select which worksheet within the Excel® file it is in, as shown here.



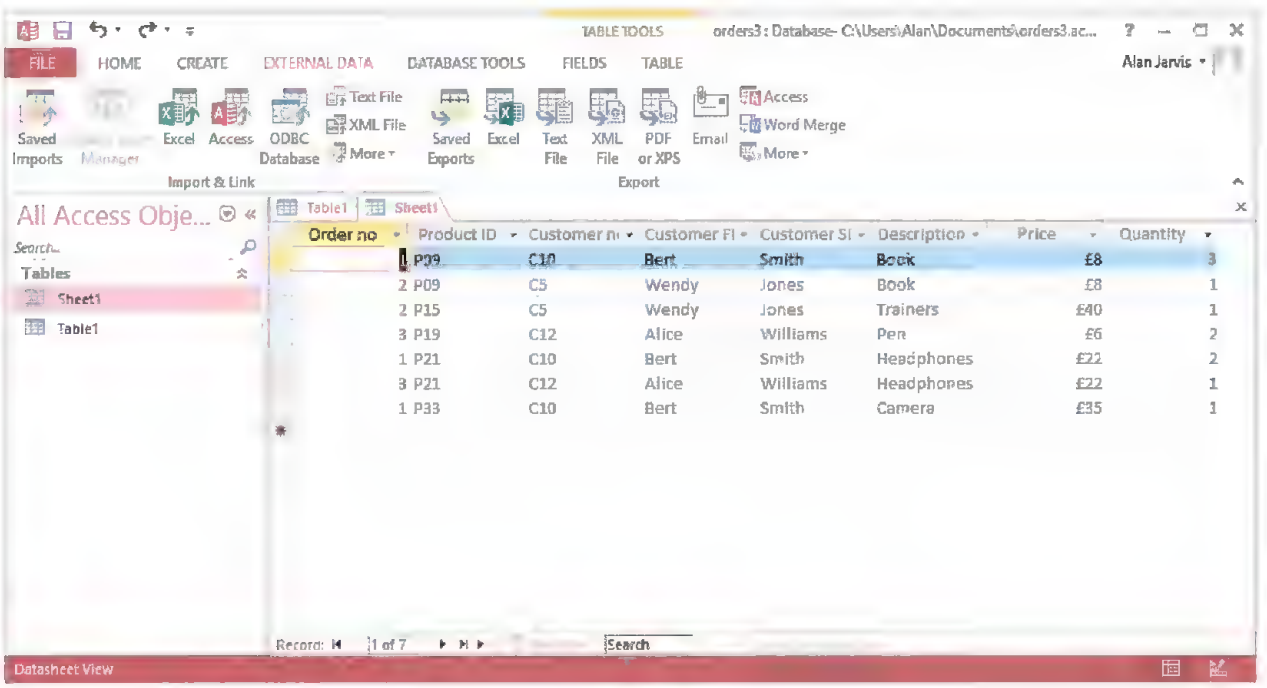
4 The next screen will ask you to confirm that the first row contains the column (field) headings, which it does in this example, so just click Next. Then you have the opportunity to adjust the field names and datatypes of each field by clicking in the field and adjusting the setting at the top, as shown here. When you are happy with the names and datatypes click Next.



5 Access® will then offer to add a primary key field to the data. Since this data will be split up into different tables, there is no need for a primary key at this stage so select the 'no primary key' option, as shown here. Then click Next.

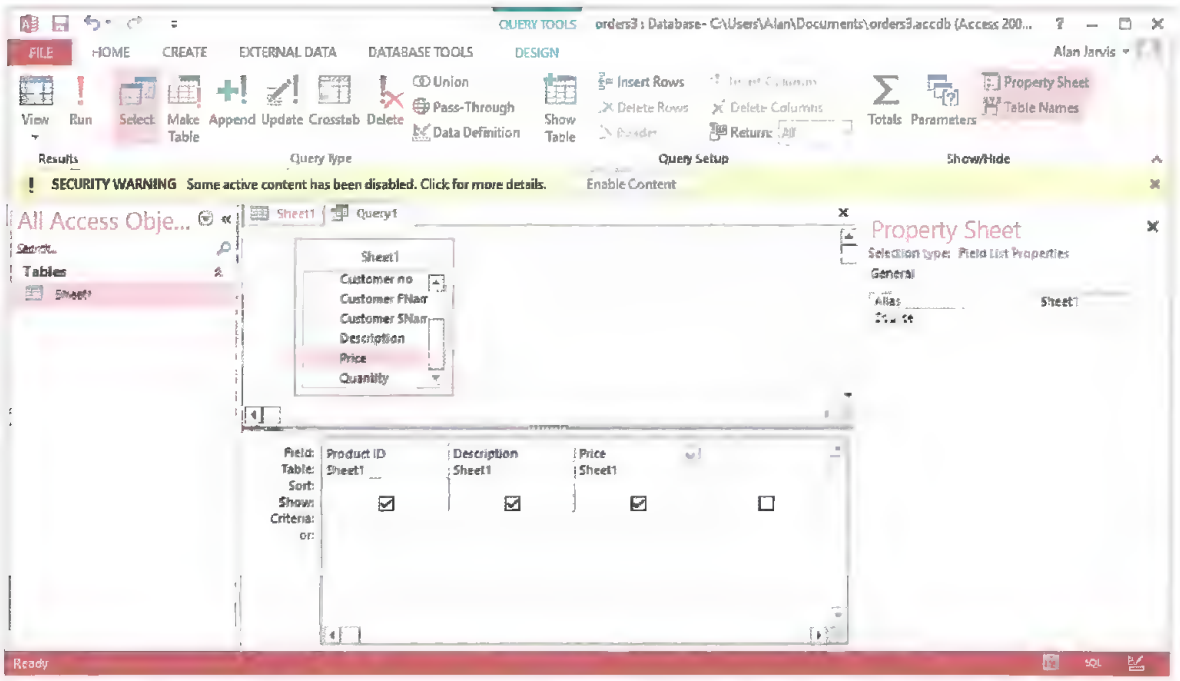


6 Finally, it will ask you which table to import the data into. As this is just a temporary table it is fine to go with the default name that Access® suggests. Click Finish to import the data. Access® asks you if you want to save the import steps, but there is no need to do this; just click Close. All the data should now be imported to a single table, as shown here.

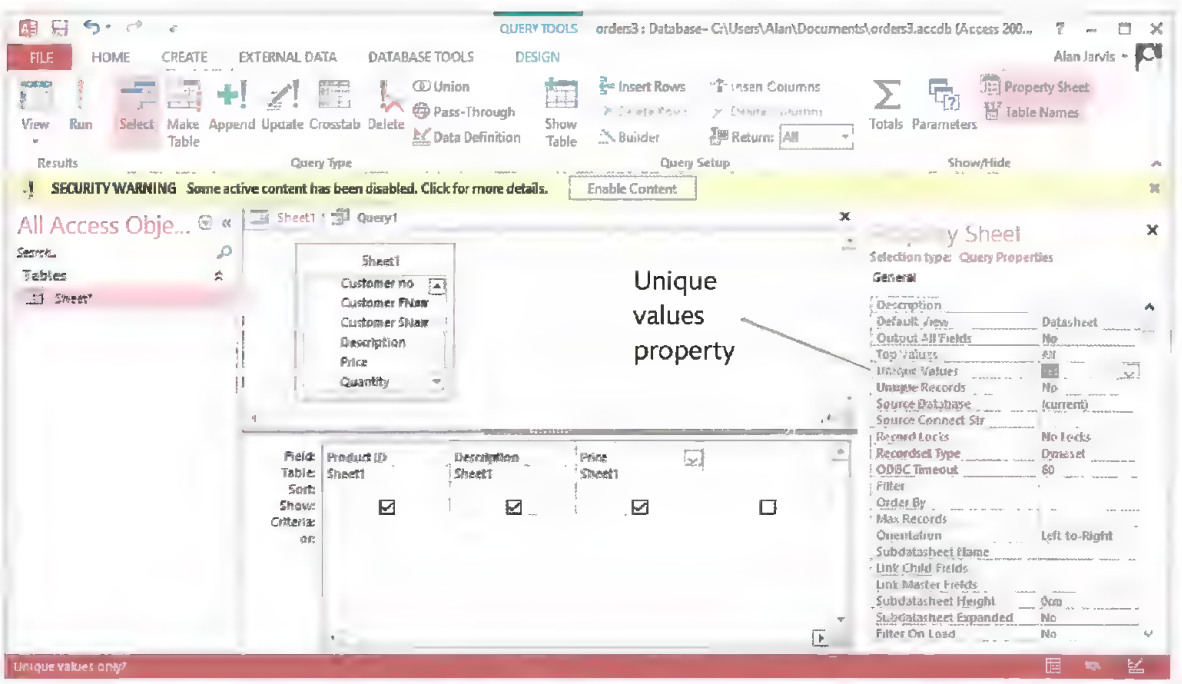


7 Now that the data is imported into Access®, we can use action queries to split it into the required tables. It is a good idea to start off with a standard select query and make sure that it has selected the correct records before you convert it into a make table query.

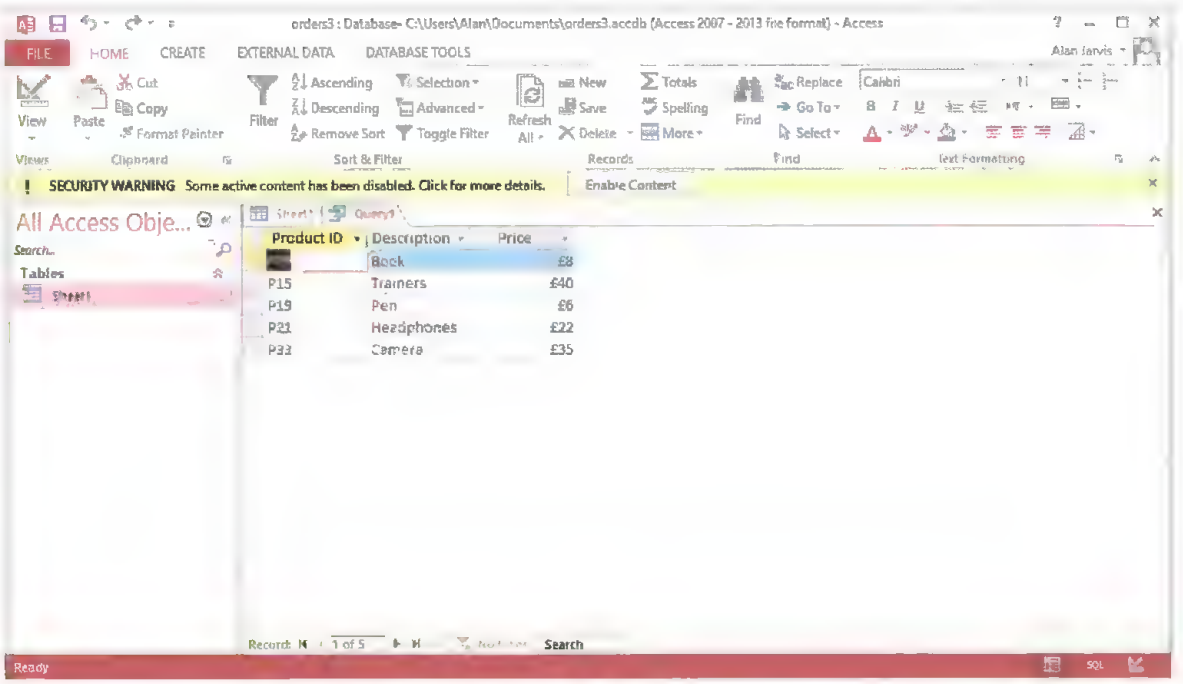
To create the Products table, first go to the Create menu and click the Query Design icon, and then add the table created from the imported data. For the Products table, the only fields we need are Product ID, Description and Price. Drag these into the query grid, as shown here.



If you run this query, you will see that some products appear multiple times (because they have been ordered on different orders). You need to create the product table so that each product only occurs once in the query results. To show only unique values, you need to click in the light blue background in the top part of the query screen, which will display the query properties in the right side panel. The fourth property in the list is called Unique Values and, by default, it is set to No. Change this to Yes, as shown here.

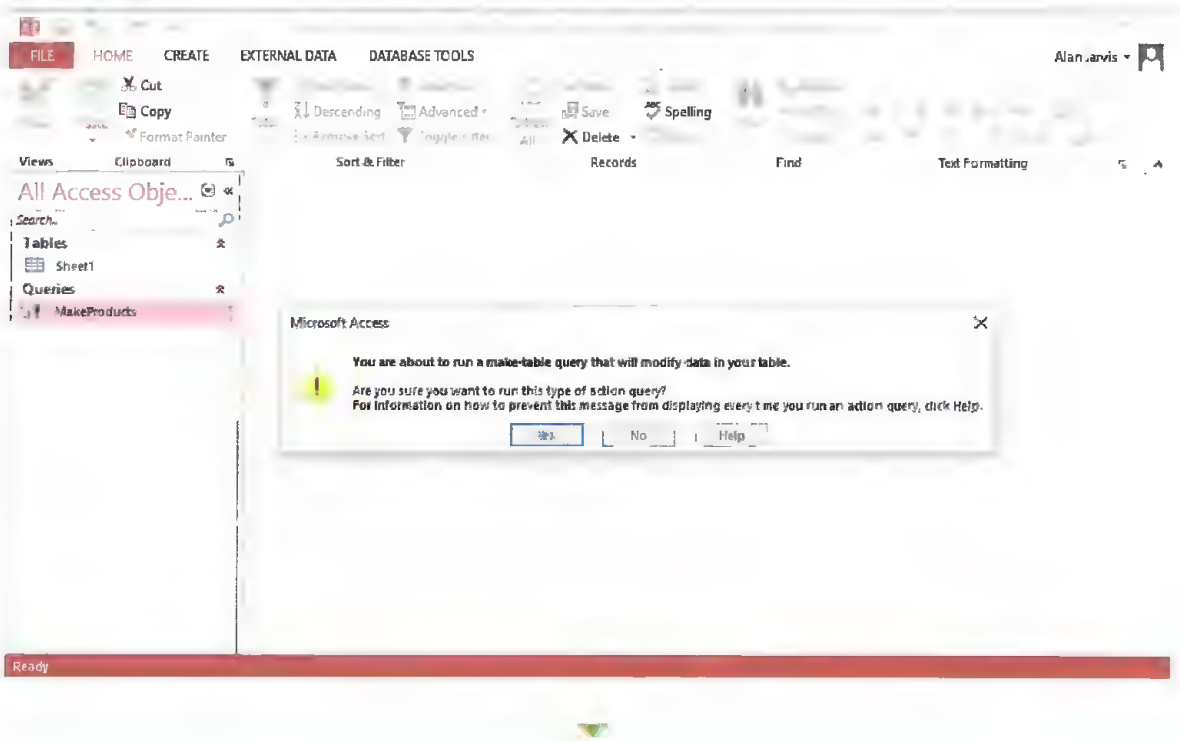


10 Now when you run the query you will only get each product displayed once, as shown here.



11 This query can now be converted into a make table query and used to create the products table. To do this, click the Make Table icon in the Design toolbar. Access® will then ask you the name of the table that you want to create. In this case it is Products, as shown here.

12 Before you can run the query in Make Table mode, you will need to click the Enable Content button below the toolbar and save the query. Open the query again and a warning message will appear telling you that the query will modify data, as shown here.



13 Click Yes. Another message will appear telling you that you are about to paste five rows (records) into a table. Click Yes again. Then your new Products table will be created. Only run the query once. You will also need to go into the Design View for the new Products table and make the Product ID the primary key for the new table. The other tables for the database can be created in the same way.

Wild cards

Queries can also use so called 'wild cards' for their criteria. A wild card is a special character which can be substituted for any other character(s). In Access® queries, there are several wild card characters as follows.

- To substitute any number of characters use '*': so North* will find North West, North London and Northend.
- To substitute a single character use '?': so B?ll will find Bill, Ball.

To substitute a single digit use '#': so 1#2 will find 123, 103.

Reports

Reports are produced to show the outcomes of queries and can be printed to present the data to users.

Worked example: Dissford Arts Centre 6

As the final part of the process of finding and then booking an available room at the arts centre, a printed confirmation sheet needs to be produced to be sent to the client as a reminder of their booking. This is done using an Access® report. The design for the report is shown here.

Booking Report	
Booking slot ID	nnnn
Session	
Slot Date	nn/nn/nn
Room name	n
Client name	xxxxxxxxxxxx
Client Tel	xxxxxx
Client address	xxxxxxxxxxxxxxxxxx
Client town	xxxxxxxx

Test plans

Testing that a completed database meets the user requirements and works correctly is an important step. To do this you will need to create a test plan which shows each individual test that you will carry out and the expected outcome of each test. Your test plan should be created at the design stage and then used to test the database once it has been developed.

Correctness of data

It is important to ensure that all the data input into a database is correct. You must ensure that the data validation that has been defined in the data dictionary works correctly and that only data that meets the requirements for each table/field is accepted, and that data which does not meet the requirements is rejected with a suitable error message.

When creating the test data to be included in the test plan for each table, there are three main types of test data to be created:

- Normal – This is data which can be considered the ordinary/typical (normal) type of data you would expect to be input to the field and that should be accepted.
- Erroneous – This is data that is incorrect for this field (e.g. a text value in a number field) and should be rejected with a suitable error message.
- Extreme – This is data which is at the limit of what should be accepted. This might be in terms of the maximum and minimum acceptable values in a numeric field, or a value with the maximum number of characters for a text field. Extreme values (for test data) should include those on the limit of what is acceptable (and so should be accepted) and just over the limit (and so should be rejected).

For example, here is the data dictionary for the Rescue dogs table.

Table 2.22: Data dictionary for the Rescue dogs table

Field name	Key?	Datatype	Validation
Dog_ID	Primary	Integer	Automatically created by Access
Dog_name		Text, 15 characters	
Breed		Text, 15 characters	
Colour		Text, 15 characters	
Sex		Text, 1 character	M or F only
Year_of_birth		Integer	Must be less than current year and no more than 20 years in the past
Approx_size		Text, 6 characters	Small, medium or large only

Here is an example of suitable test data for each of the fields which have validation applied (assuming that the current year is 2016).

Table 2.23: Suitable test data for each of the fields of the Rescue dogs table

Field	Test data type	Data item	Expected outcome	Actual outcome (completed when the database is tested)	Action required?
Sex	Normal	M	Accepted		
Sex	Normal	F	Accepted		
Sex	Erroneous	X	Rejected		
Year_of_birth	Normal	2010	Accepted		
Year_of_birth	Erroneous	Text	Rejected		
Year_of_birth	Extreme	2016	Accepted		
Year_of_birth	Extreme	2017	Rejected		
Year_of_birth	Extreme	1996	Accepted		
Year_of_birth	Extreme	1995	Rejected		
Approx_size	Normal	Small	Accepted		
Approx_size	Normal	Medium	Accepted		
Approx_size	Normal	Large	Accepted		
Approx size	Erroneous	Big	Rejected		

This gives examples of the different types of test data that could be used. However, where a field has a limited number of acceptable entries (such as the Sex and Approx size fields) the best method of ensuring that only valid data is entered is by using the Lookup wizard, as described earlier.

Link

For more on using the Lookup wizard see Lookup tables.

Functionality

The user interface needs to be tested to ensure that all of its features, including forms and reports, work correctly. This is called testing the user interface's functionality.

Testing the functionality of a user interface is rather more complex than testing the data. You need to create a series of scenarios which mimic the way that users will use the completed database system. (You need to include, in these scenarios, data which is normal, extreme and erroneous, where appropriate.)

Consider the database design created earlier for the process of searching for available booking slots and then making a booking in the Worked example: Dissford Arts Centre. Booking scenarios for this process would need to have some existing data in the database, including clients and available and booked booking slots.

When booking a room, the data that is entered into the database system is the date range of available slots to search for (start and end dates) and the client ID. So test data which is normal (for example, a client ID who does exist) and erroneous (for example, a client ID who does not exist) needs to be created.

An example of the test data that could be used is shown below. Note that the date ranges will depend upon the example data which has been loaded into the database.

Table 2.24: Test data for Worked example: Dissford Arts Centre booking database

Field	Test data type	Data item	Expected outcome	Actual outcome (completed when the database is tested)	Action required?
Date range	Normal	1/11/15 to 10/11/15	Accepted, produces list of free slots		
Date range	Erroneous	20/12/15 to text	Error message		
Date range	Erroneous	Date in the past to date in the past	Error message		
Client ID	Normal	2 (valid client ID)	Accepted, client ID saved in the booking slot table for the date selected		
Client ID	Erroneous	999 (invalid client ID)	Rejected with error message		

Usability and accessibility

The user interface can also be tested for usability and accessibility.

- Usability – Usability testing is rather more subjective than functionality testing and tests need to be conducted by real users who then give their opinion about how easy the interface is to use and any suggestion for improvements. This cannot really be done at the design stage. Usability tests need to be conducted using either a prototype of the user interface or the completed version.
- Accessibility – The accessibility of the user interface needs to be tested against existing guidelines for making computer interfaces accessible, such as the Royal National Institute of Blind People (RNIB) guidelines for web accessibility. These guidelines will give consideration to things such as font size and colour and the use of white space.

Link

The RNIB guidelines for web accessibility can be found at http://www.rnib.org.

PAUSE POINT

How much testing is enough? Do you think it would be ok to just check that the basic things work and then give the database to the users for them to find any faults for you to fix?

What might the consequences of this approach be?

Testing is a big subject with many different aspects to it. Research different types of testing and consider how they might be applied to a database system.

C Creating a relational database structure

Once the design of a database is complete, you can start work on developing the relational database solution.

Producing a database solution

In this section, you will learn about selecting and configuring appropriate RDBMS and SQL tools to produce a database solution to meet the client's requirements.

Creating data tables, relationships and validation rules

Using the data dictionary developed during the design stage, the database tables can now be created. This can either be done by defining the fields, datatypes and validation rules in the Access® table Design View or, if the tables will use imported data, this can either be done at the import stage or using action queries to split up a single imported file. Once the tables have all been created, you can then create the relationships between them.

Link

Creating data tables

Details of how to create tables in Access® is covered in the Step by step: Creating tables.

Creating links and relationships between data tables

Details of how to create relationships in Access® is covered in the Step by step: Creating relationships in Access®.

Applying data validation rules

Details of how to create lookup tables and validations rules is in the Step by step: Lookup tables and validation rules.

Worked example: Dissford Arts Centre 7

Tables can be created for this booking database using the data dictionary and ERD created earlier. The Design View of the Client table is shown in Figure 2.12 on the following page.

An AutoNumber datatype is used for the Client_ID so that each client will have a unique number automatically created by Access®.

(Note also that the Client_tel is a text field, as a telephone number is not a number in the mathematical sense (e.g. you never need to add phone numbers together!) and using a numeric data type will prevent you from adding leading zeros, which most phone numbers require.)

Remember that when you use a foreign key in another table, such as the Client_ID in the Booking_Slots table, you must use a datatype that matches the primary key in the linked table. In this example, as the Client_ID is an AutoNumber field, so you must use the data type of Number, long integer.

Once the tables are created, you can create the relationships between them using the Relationships window, as described earlier. The relations set up between the tables for the arts centre booking system are shown in Figure 2.13 on the following page.

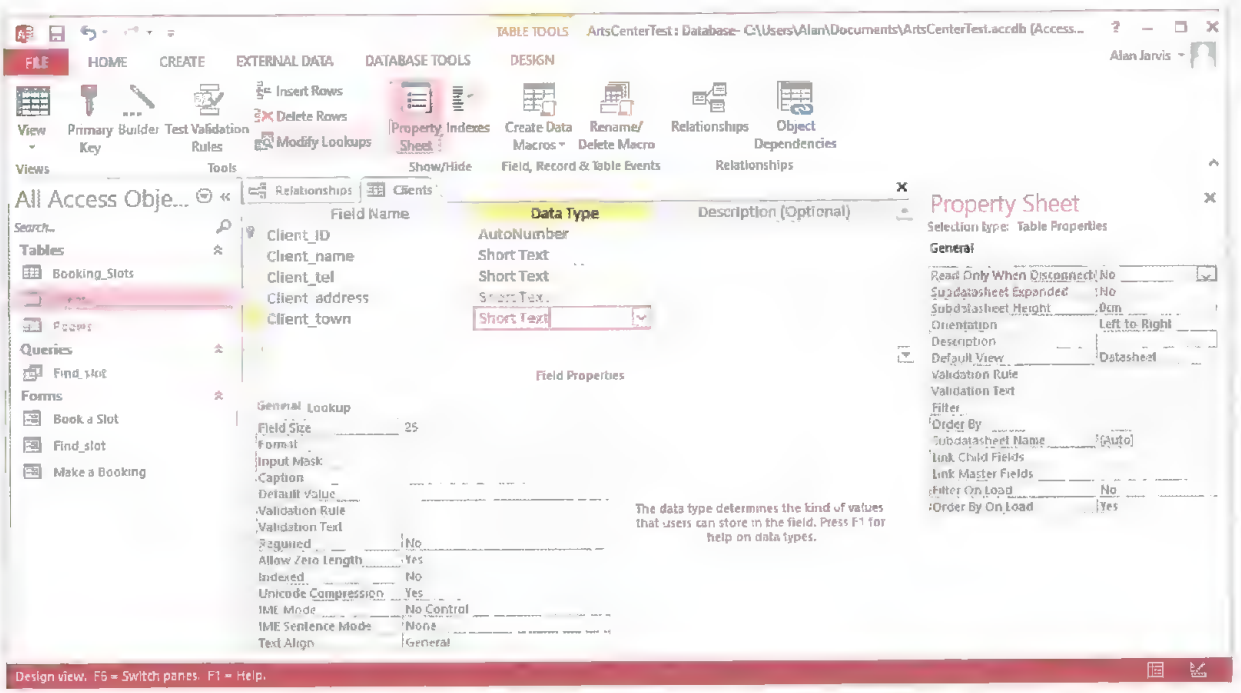


Figure 2.12: The Client table design

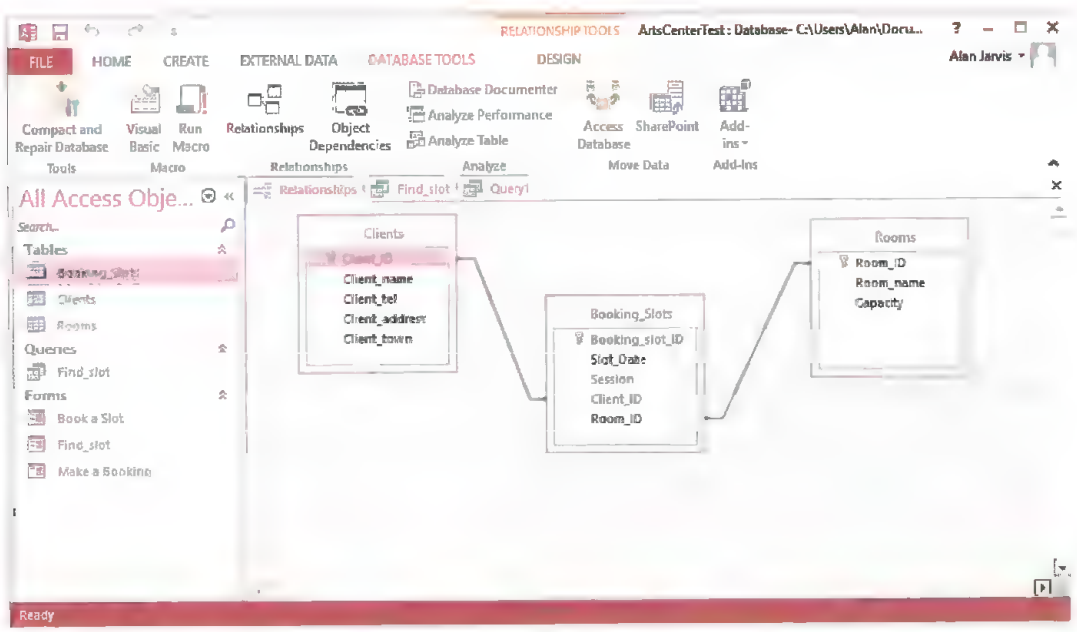


Figure 2.13: The relationships for the arts centre database system

At this stage, you should enter a small amount of test data into the tables, just to ensure that they work as expected and to provide something for you to test queries and forms and to report with. It is unwise to enter large amounts of data until you are sure that the database works as expected. If the data has already been provided for the table, now is the time to import it and, if necessary, divide it up into the required tables using action queries, as described earlier.

Worked example: Dissford Arts Centre 8

An important concept in a booking database of any type (for hotel rooms or airline tickets, for example) is that of available booking slots. With the Online orders database, when no orders are placed, then there are no records on the Order table, but with the bookings database the concept is a bit different and you need to populate the Booking_Slots table with unbooked slots to allow users to search for and book available slots.

Typically, this would be done for three- or possibly six-month periods in advance, and in a real database system they would probably be generated automatically. An easy way to create booking slots for the arts centre system is to use a spreadsheet as it is easier to copy and paste data in a spreadsheet and the autofill feature will create dates for you.

You will need to create an afternoon and evening session booking slot for every room on every date. This creates quite a lot of records (8 per day), as shown in Figure 2.14.

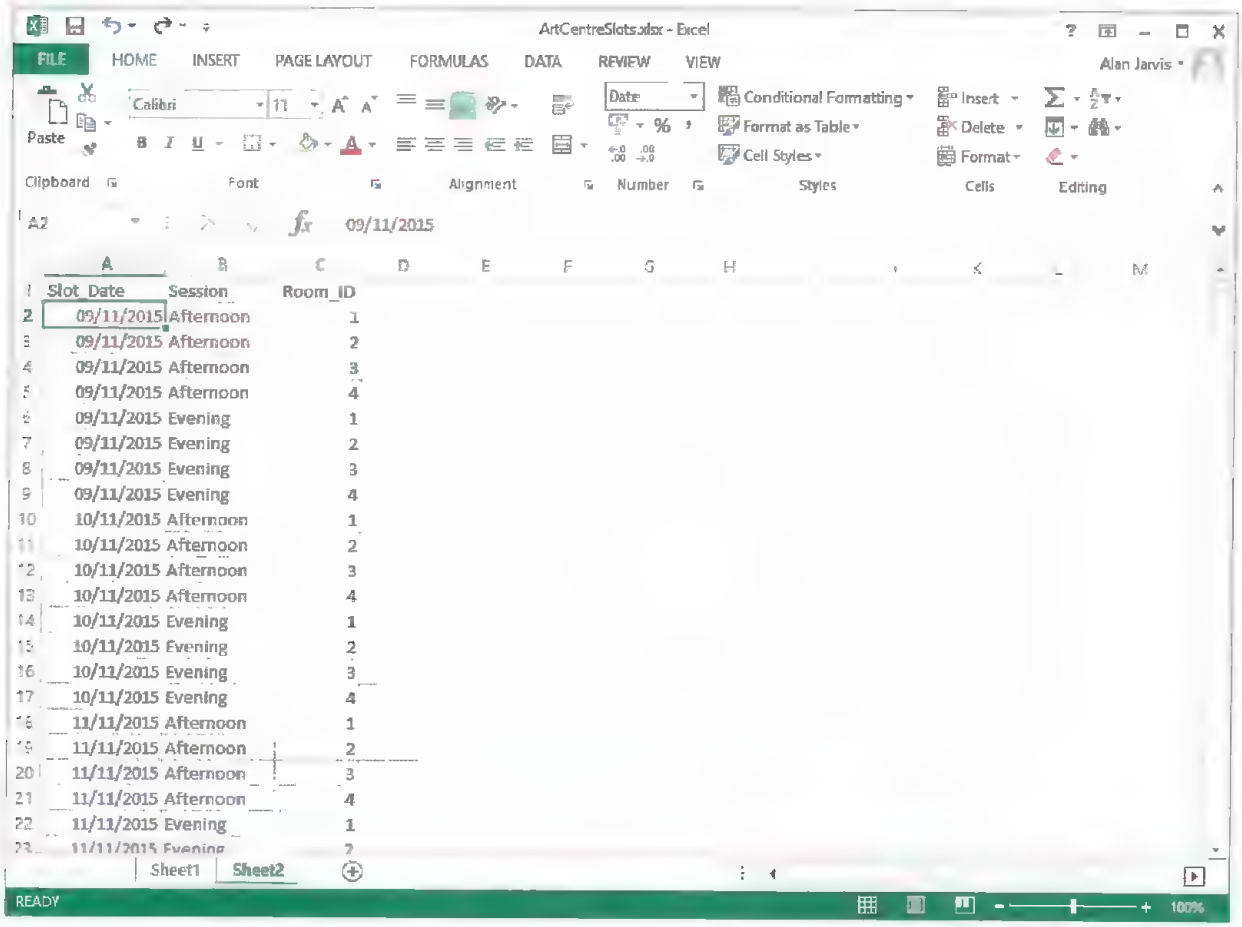


Figure 2.14: Spreadsheet data for the Booking Slots table. Once you have created the data in Excel®, you can import it into the Booking_Slots table in Access®.

Generating outputs

The next step in developing the database is to create queries. In many cases, forms and reports rely on queries to provide their data, so you can start developing the queries before the forms and reports. However in some cases, such as where a query takes its criteria from a form, you may need to develop them together.

User generated queries

Database users can, of course, generate their own queries to answer *ad hoc* questions about the data, but many queries will run automatically in the background, triggered by the forms and reports that use their data.

Worked example: Dissford Arts Centre 9

You have already created a design for a query that will find available booking slots. Now you can start to implement that design. First, you will create the basic query to find available slots, later you will integrate it with two forms to provide a facility to find and then book an available slot.

The data shown in the table in Worked example: Dissford Arts Centre 2 has been added to the tables that provide both booked and available slots, to allow us to test the query. Figure 2.15 shows the query design for this.

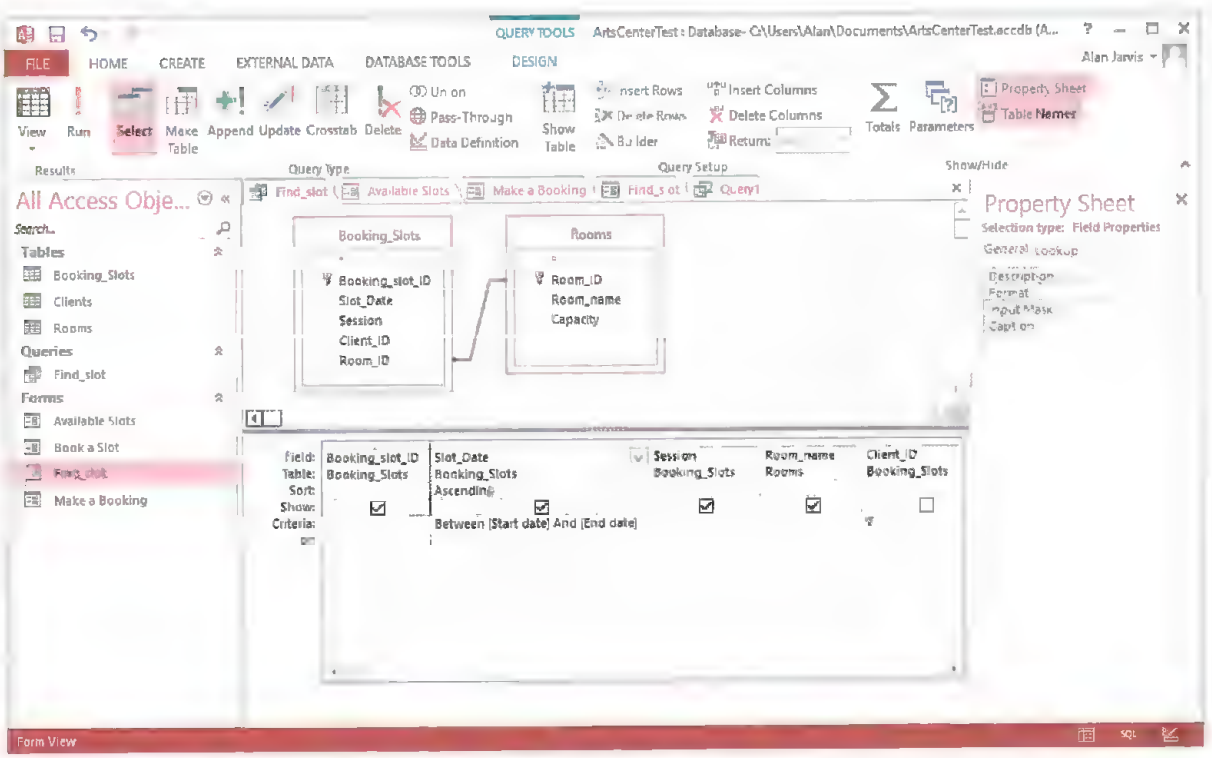


Figure 2.15: Query design

The criteria added for the Slot Date field will display a dialog box asking for the start date, then another asking for the end date of the search range. Slots will appear in ascending order (earliest first). With the criteria for the client_ID set to zero, only those booking slots with no client ID (i.e. available slots which are not booked) will appear. Because the 'Show' option is not selected, this will not appear in the query results. Figure 2.16 shows the typical output from this query with a date range set to between 9/11/15 and 10/11/15.

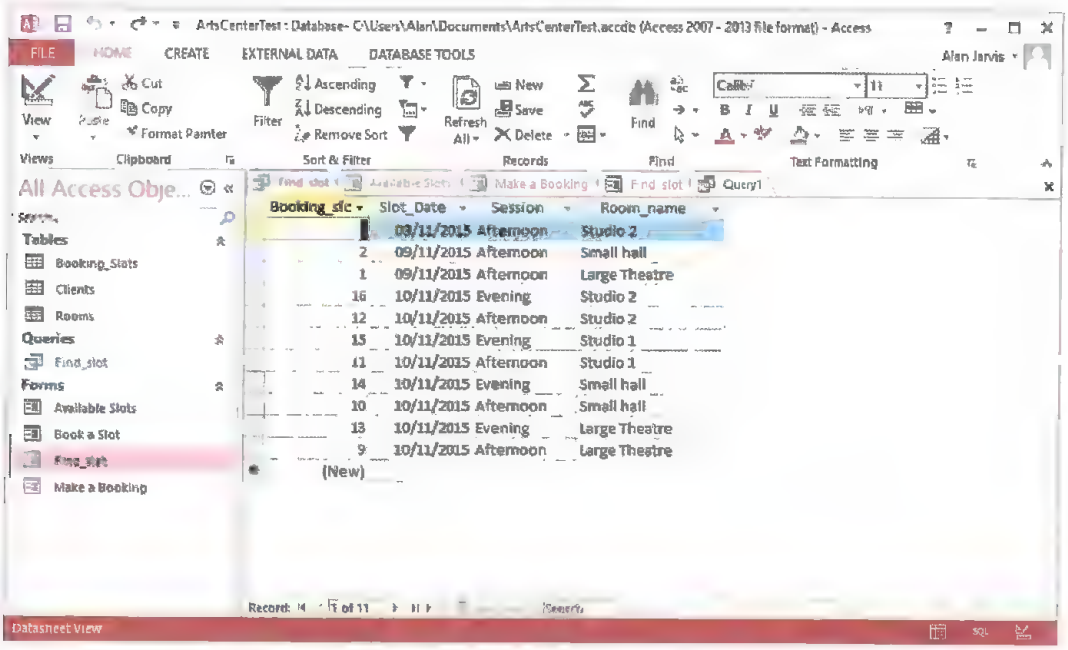


Figure 2.16: Typical query output with a date range set to between 9/11/15 and 10/11/15

The query that you have just created will not be directly visible to the user. Instead, it will work while being automatically hidden behind the form that the user will see. The next step is therefore to create that form.

Worked example: Dissford Arts Centre 10

Creating a form that will list all the available slots that the query has found is quite straightforward if we use the Form Wizard icon (found in the Create toolbar). Start from the Design View of the query that you have just developed; the form will automatically be based on it. Select all the fields from the query, as shown in Figure 2.17.

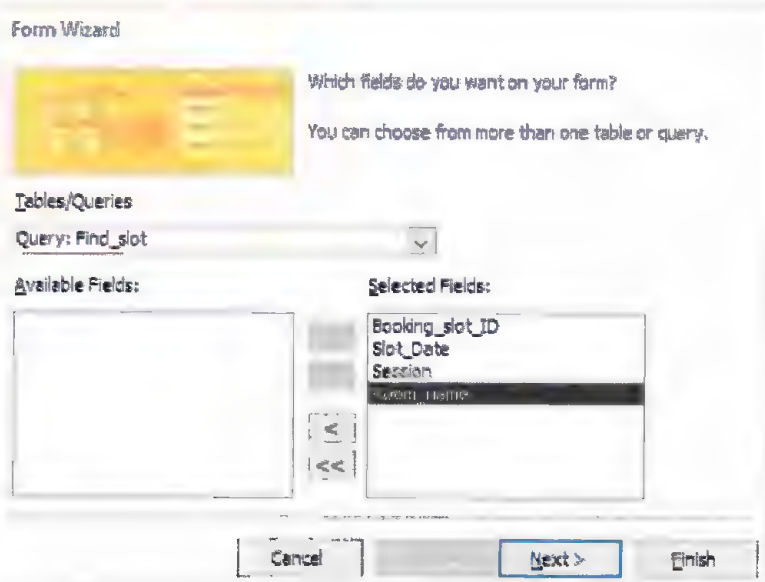


Figure 2.17: Selecting the fields for the form

Then click Next and choose a Tabular layout so that it displays many records on a single form. Click Next. Choose a title for the form such as 'Available Slots' and then click Finish. Because the form is based on the query, it will now run the query and ask for start and end dates. Enter these and you should see the data from the query displayed in the newly created form, as shown in Figure 2.18.

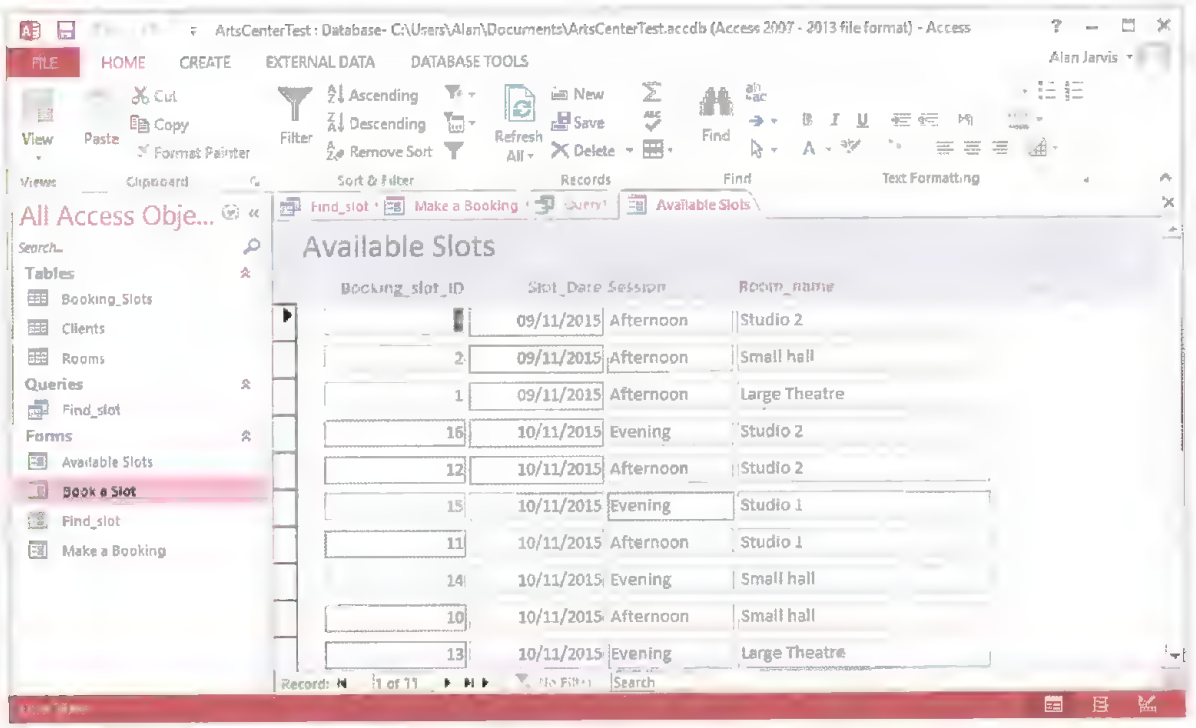


Figure 2.18: The completed form

Now the first stage in the process is complete. The user can enter dates and see a list of available booking slots. The next stage is to allow the user to select a particular booking slot and make a booking for that slot.

Worked example: Dissford Arts Centre 11

First, a simple query is required to select the record in the Booking_Slot table that has been chosen (i.e. the one the client wants to book). This query is very similar to the Find_slot query just created (in Worked example: Dissford Arts Centre 10) but it needs to pick up the Booking_slot_ID that the user has selected from the form that has just been created. The criterion required to pick out that record is shown in the query design for the new query, in Figure 2.19.

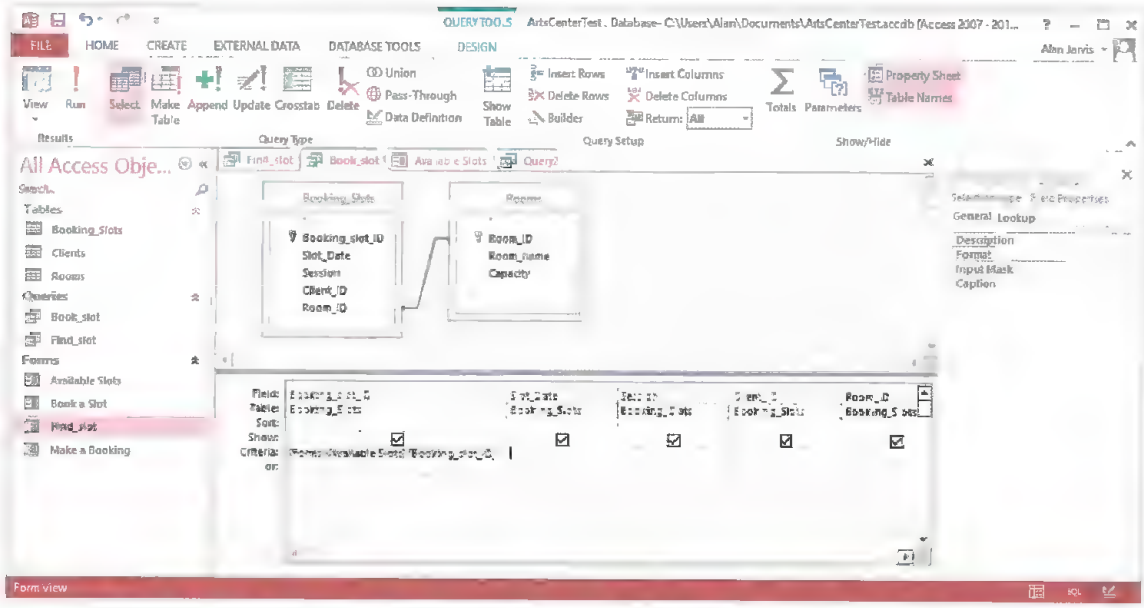


Figure 2.19: Criterion for the Booking_slot_ID

This query is now saved. In this example, it has been give the name Book_slot.

You now can create a form which will display the data from the Book_slot query. Open the query in Design View and then, from the Create menu, choose the Query Wizard. Make sure it is using the Book_slot query and select all the fields (see Figure 2.20). Click Next.

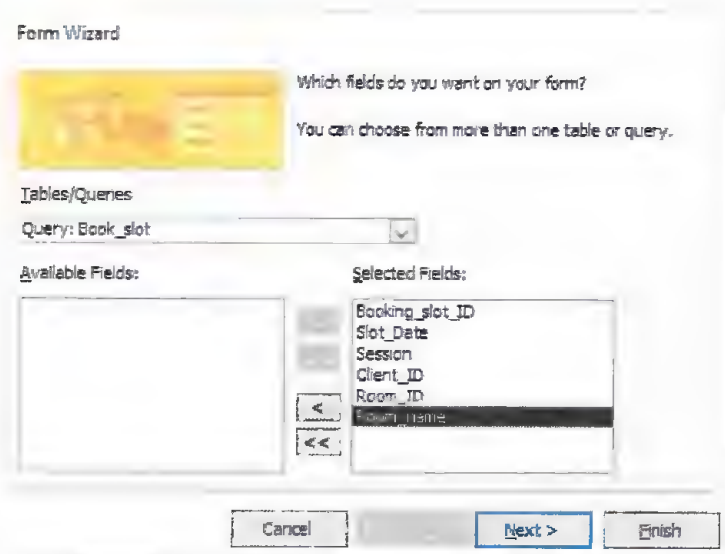


Figure 2.20: Selecting the fields

This time, use a Columnar layout (as only one record will be displayed in the forms – the one the user has chosen to book) and give the form a title such as Make_Booking.

If you complete the Form Wizard and the new form attempts to open, you will get an error message because the query that provides the data for the form is looking for a value for the Booking_slot_ID from the Available Slots form. The new Make_Booking form does not work on its own – it only works in conjunction with the Available Slots form. Therefore, you need to modify the Available Slots form to link the two forms together. Click Cancel on the error message and close the form.

Now return to the Available Slots form and open it in Design View, as shown in Figure 2.21. You are going to add a command button which will open the Make_Booking form.

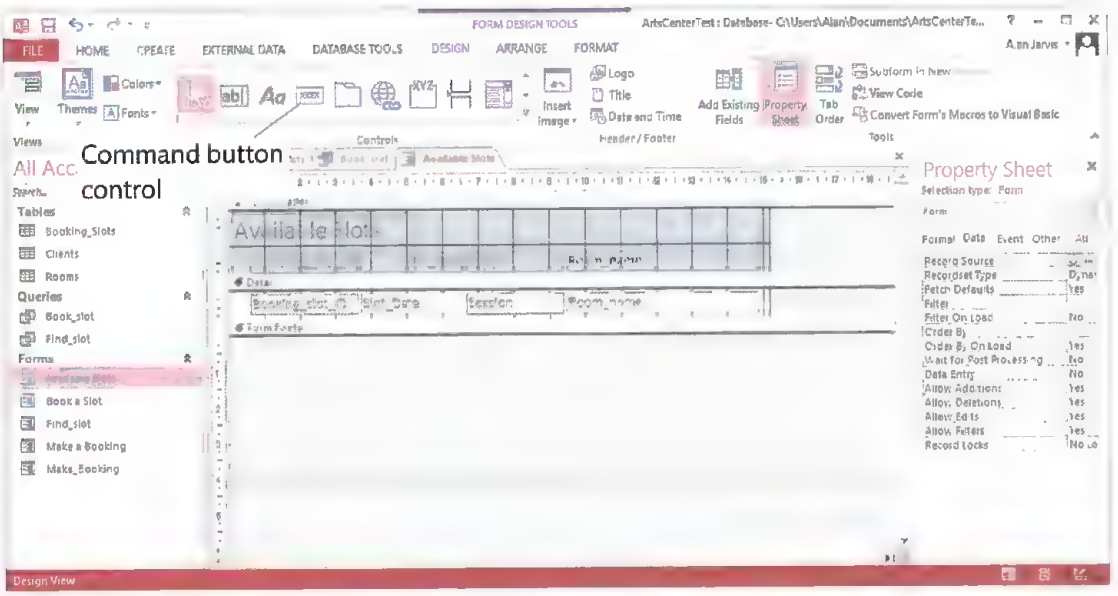


Figure 2.21: The Available Slots form in Design View

Make sure that you have Design View selected in the menu bar, and, in the Controls section of the toolbar, you need to click on the Button control and then drag out a button on the form design to the right of the Room_name field. This will open the Command Button Wizard, as shown in Figure 2.22.

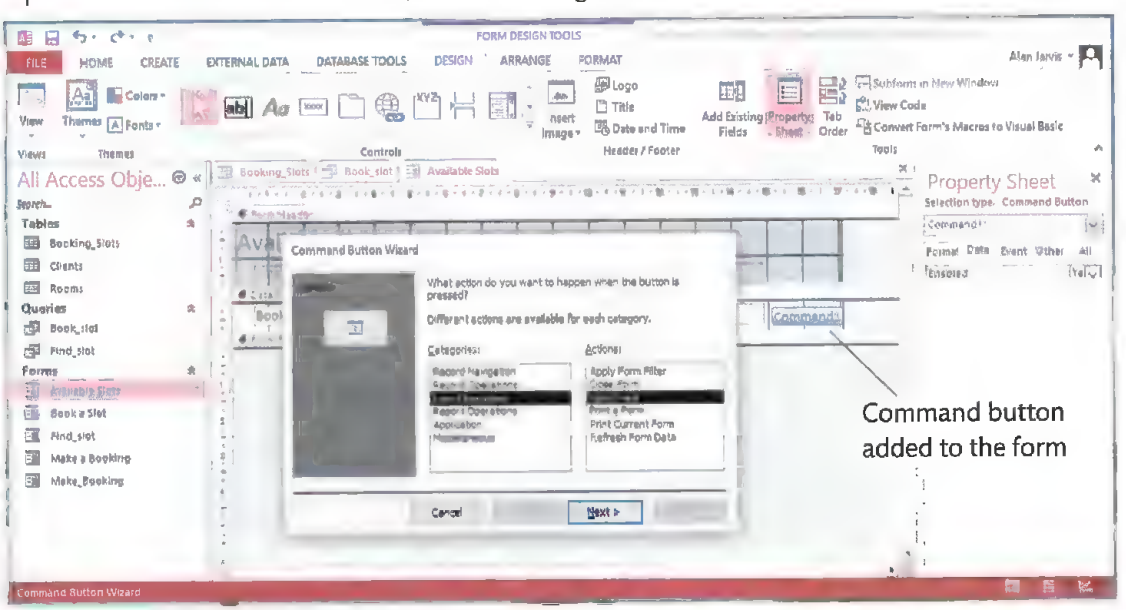


Figure 2.22: The Command Button Wizard

From the Categories section choose Form Operations, and then, in the Actions section, choose Open Form. Click Next. Now choose the Make_Booking form as the form to be opened, click Next and in the next step leave the option set to 'Open the form and show all the records'. Click Next again. In the next step (see Figure 2.23), set the text for the button to 'Book' and then click Next.

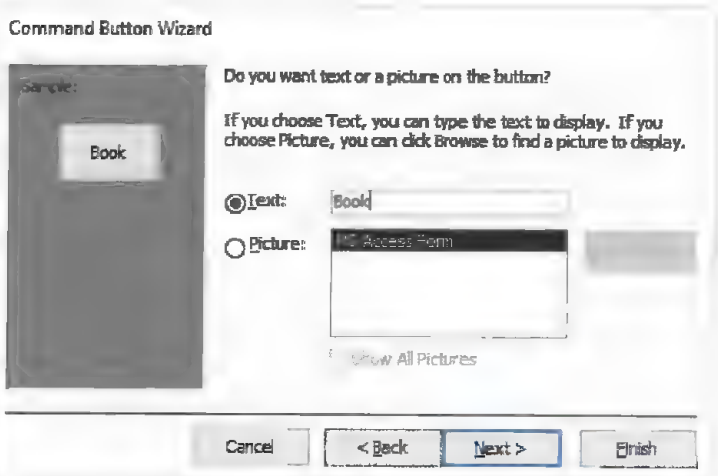


Figure 2.23: Setting the text to appear on the button

This will take you to the final step of the wizard where you can just click Finish. The form should now contain a button labelled 'Book' as shown in Figure 2.24.

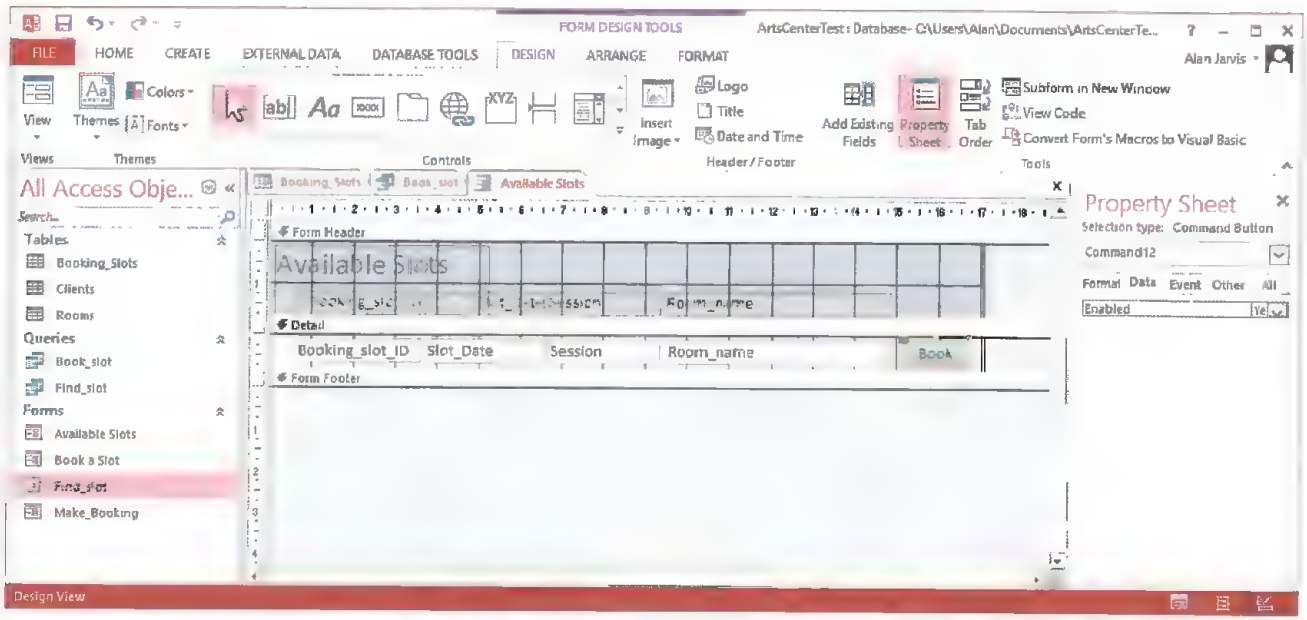


Figure 2.24: Form design with a button added

Save and close the form.

Now it is time to test the whole process of searching for an available booking slot, selecting the one to book and booking it. Open the Available Slots form and enter a valid date range. The form should open, showing all the available booking slots, with a 'Book' button next to each one (see Figure 2.25).

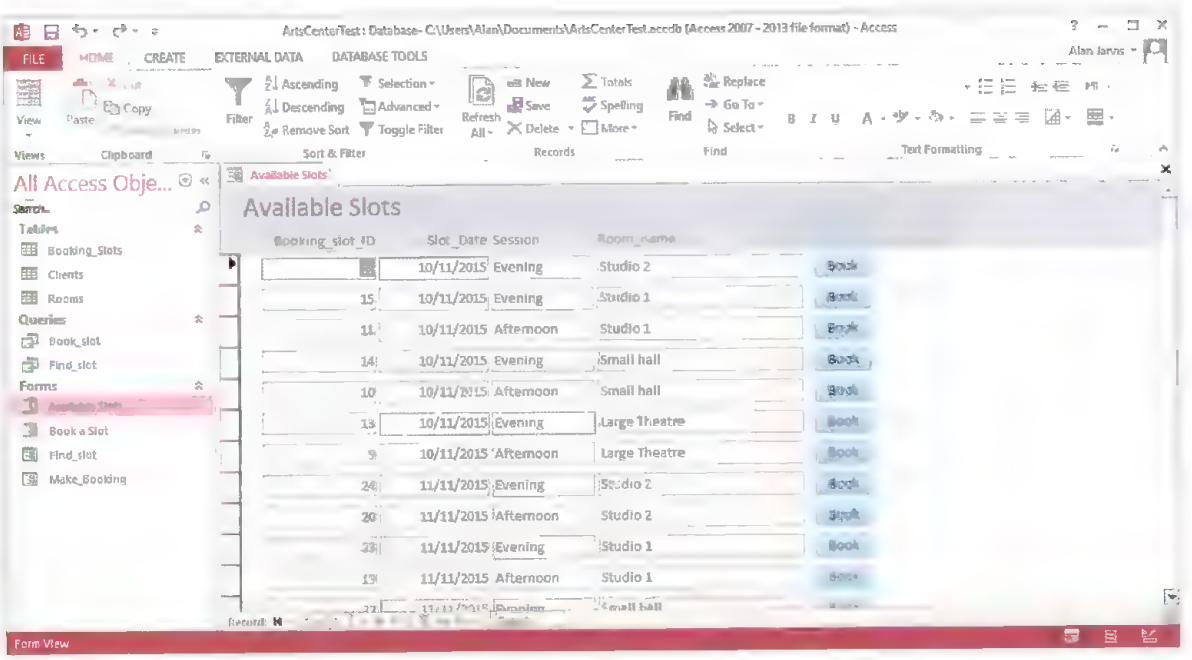


Figure 2.25: Slots available for booking

Select one slot and click the Book button. The Make_Booking form should now appear with the booking slot you chose in the previous form shown (see Figure 2.26).

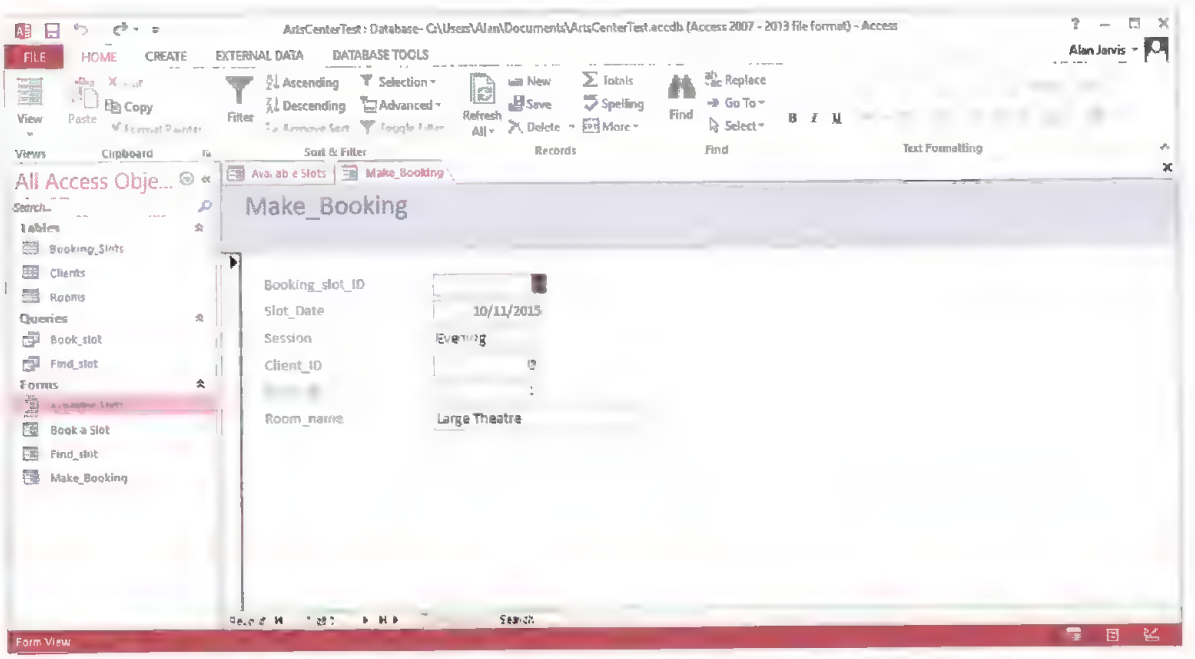


Figure 2.26: The Make_Booking form

The Make_Booking form in Worked example: Dissford Arts Centre 11 works because it displays only the Booking_Slots record that the user clicks in the Available Slots form and a booking can be made by entering a Client_ID value in that field on the form.

However it is not fit for purpose for a number of reasons.

- The user can type in all the fields of the form and change their values. The room ID and name, date, session and the Booking slot_ID must not be changed as they are related to this particular booking and changing them might create duplicate booking slots. The only field the user should be able to change is the Client_ID. This must be changed to allocate the booking slot to a client.
- As stated above, the booking is made by changing the Client_ID from zero to a valid Client_ID. But how does the user know what Client_ID relates to a particular client? They can type any number in the Client_ID field, including invalid ones.

To resolve these issues, you need to lock all the fields on the form so that the user cannot type in them, only view them, and provide a drop-down list of valid client IDs that the user can select from.

Worked example: Dissford Arts Centre 12

To select the Client_ID, we will provide the user with a drop-down list of valid Client_IDs and names from which they can select the correct one. This will prevent any invalid data from being entered.

(Note that it does require that the client exists on the database before a booking can be made for that client.)

View the Make_Booking form in Design View. Select the Client_ID field and label and delete them (right click on them and choose Delete). Now drop down the list of controls under the Design menu and choose the Combo Box control (see Figure 2.27).

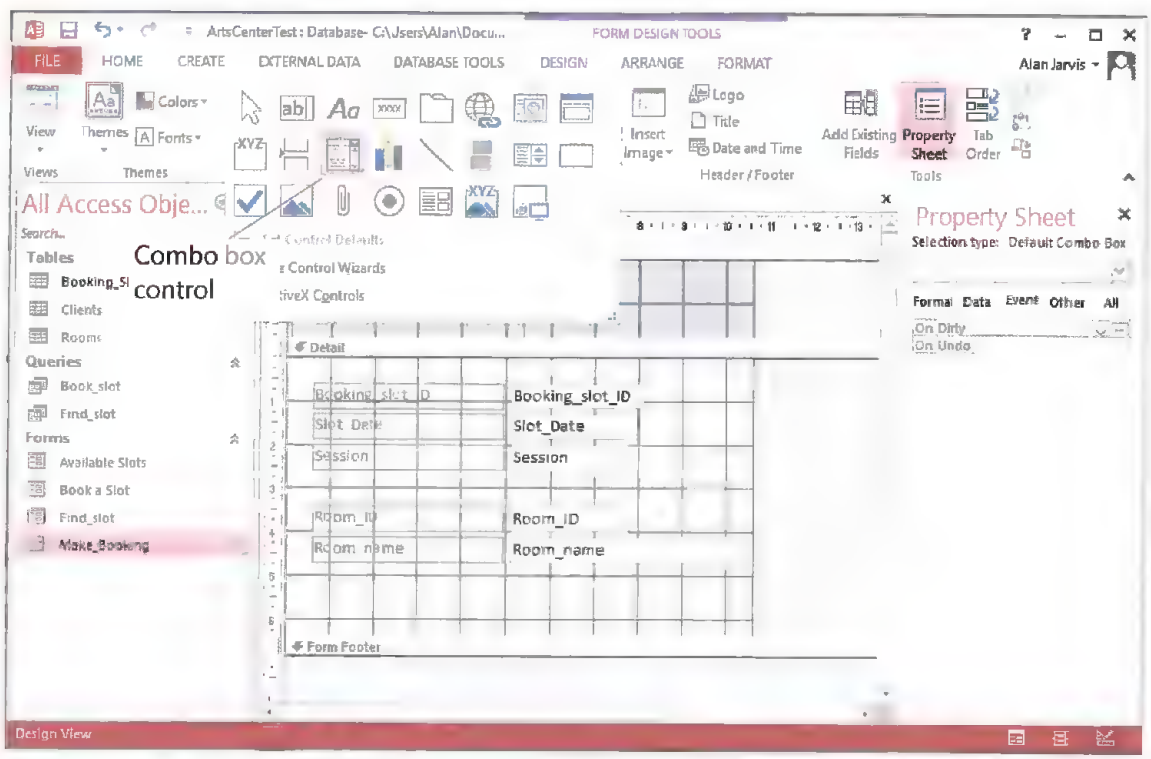


Figure 2.27: The Combo Box control

Then drag out a combo box in the space left by the Client_ID field. The Combo Box Wizard will then start. In the first step of the wizard, leave the option 'I want the Combo Box to get the values from another table or query' selected and click Next. Then choose the table called Clients as the source of values for the combo box, and click Next. Select the Client_ID and the Client_name as the fields to be displayed (see Figure 2.28).

Figure 2.28: Fields to be displayed in the Combo Box

Click Next to take you to the step where you can choose a sort order. This is not really needed here, so just click Next. Then uncheck the box which hides the key column because you need to use this field (see Figure 2.29).

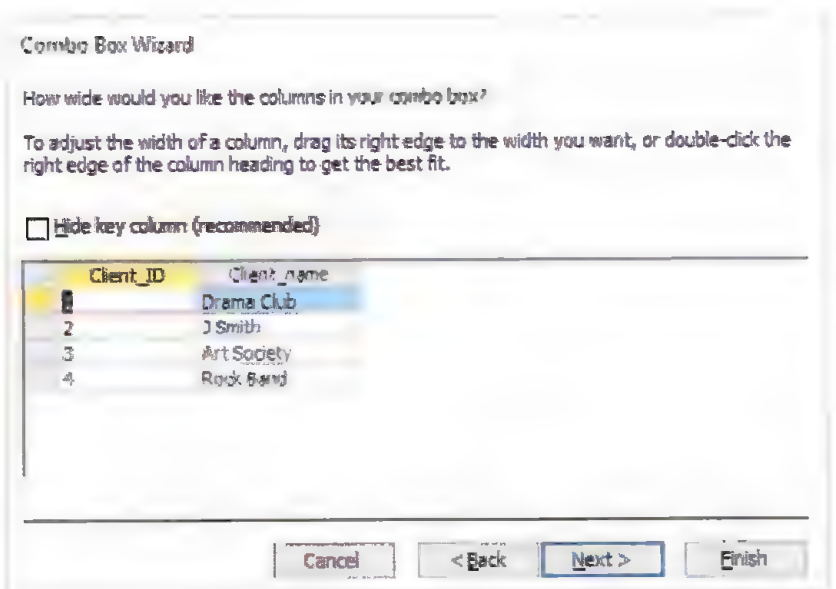


Figure 2.29: Fields to be displayed

You will now be asked which field you want to store in the database. Make sure that Client_ID is selected and click Next. This field needs to be stored in the Client_ID field in the Booking Slots table, so select this, as shown in Figure 2.30.

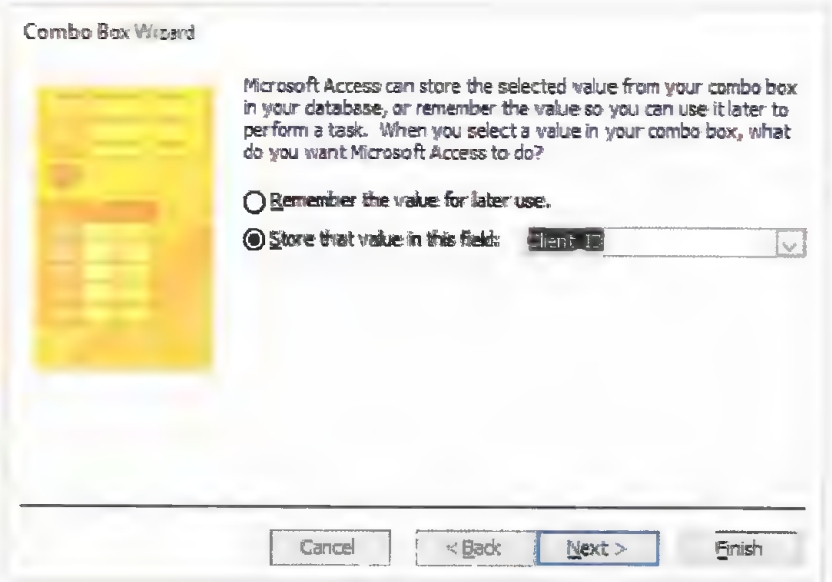


Figure 2.30: Storing the Client_ID

Click Next. Give the Combo Box a suitable title, such as Client ID, and click Finish. The form should now look like Figure 2.31.

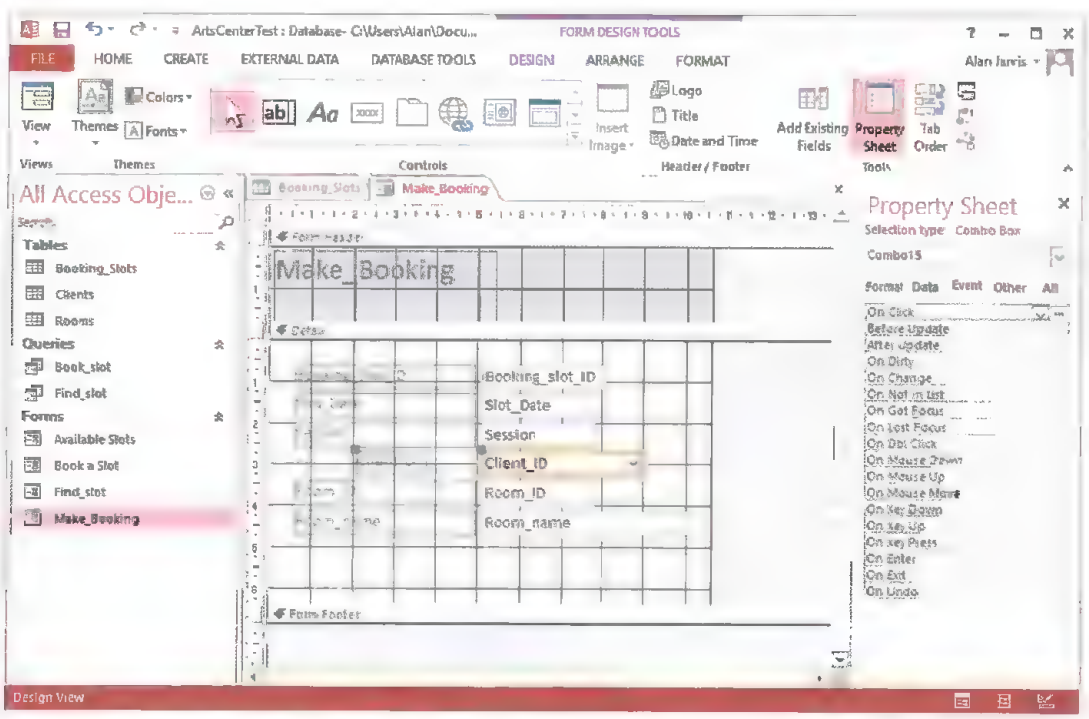


Figure 2.31: Form with Combo Box added

Finally, we need to lock all the other fields to prevent the user changing them. Click on each of the fields in turn (except the Combo Box you just created) and, on the right-hand side of the display in the Property Sheet, select the Data tab and set the Enabled property to 'No' and the Locked property to 'Yes', as shown in Figure 2.32.

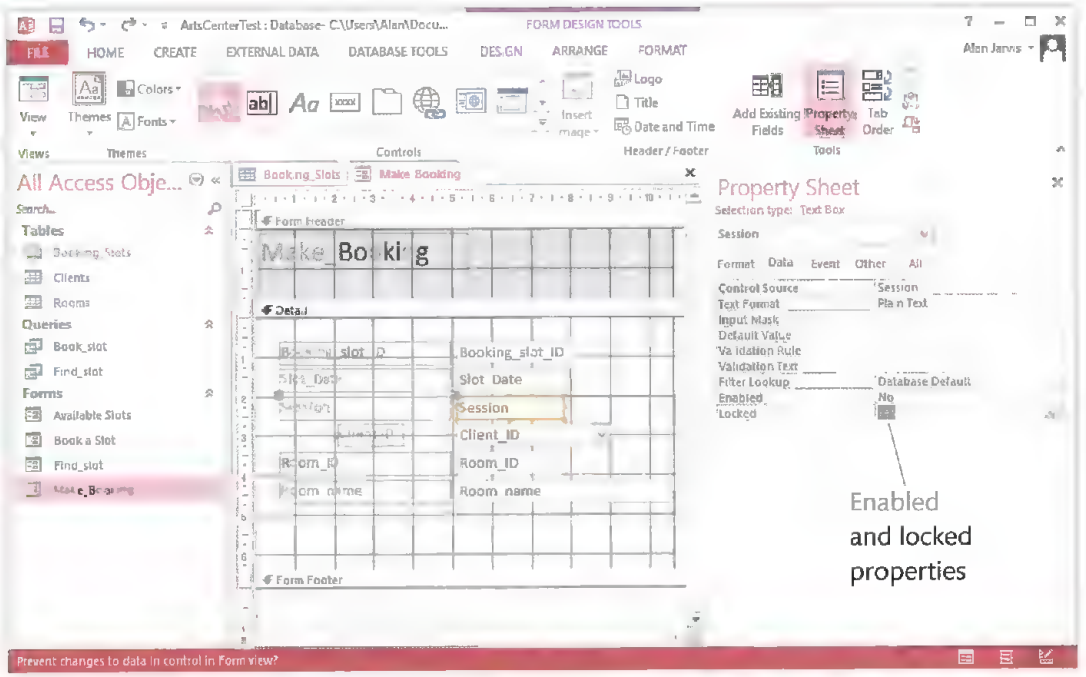


Figure 2.32: Setting the Enabled and Locked properties

One final change needs to be made to the form. The user needs an easy way, once they have selected the correct client, to be able to save the change that they have made. Therefore we will add a Save command button to provide this. Use the Command Button Wizard, as before, to add a button at the bottom of the form. Choose the Category 'Record Operations' and Action 'Save Record'. Add some text such as 'Save Booking' to the button.

Save the form and return to Form View to see the finished form. It should look something like Figure 2.33.

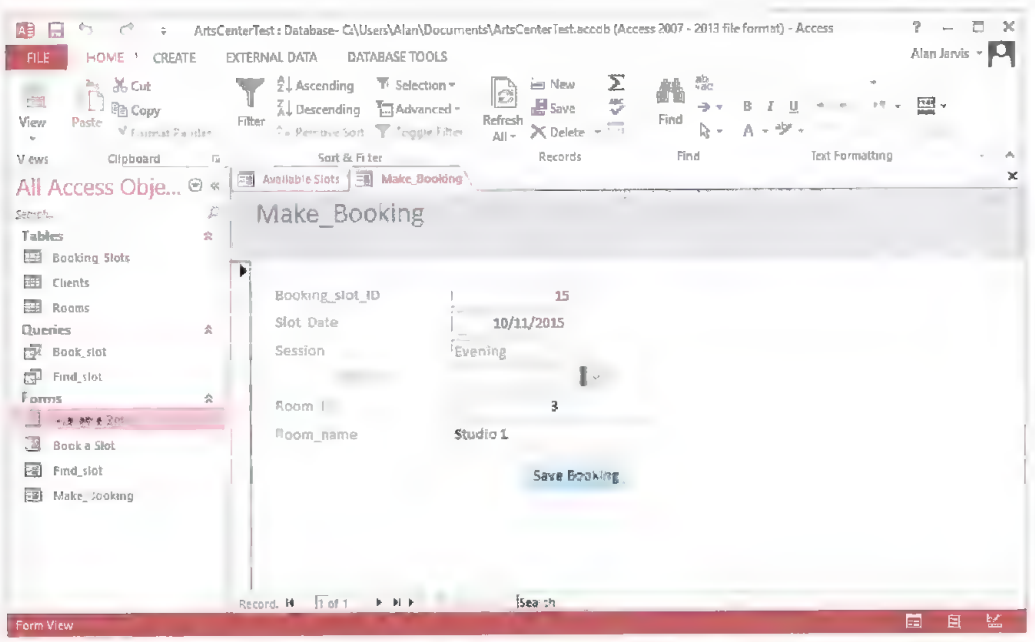


Figure 2.33: The completed form

Reports

Creating printed reports in Access® is quite easy to do because there is a step-by-step wizard provided to create report layouts. Reports can be based on either tables or queries.

Worked example: Dissford Arts Centre 13

This example creates a report in the form of a booking receipt for the Dissford Arts Centre. The first task is to create a simple query that will provide the data for the report. This needs to select a specific booking slot and then bring together the data about the client who made that booking. First, create a query in Design View, as shown in Figure 2.34.

Figure 2.34: The Query Design

Note that this query is a parameter query, which will ask the user for the booking ID.

You can check that this query works, save it (in this example it has been called Booking_report) and then move on to create the report based on it. To do this, go to the Create menu and select the Report Wizard icon. Make sure that you have the Booking_report query selected and then choose all the fields in the query, as shown in Figure 2.35.

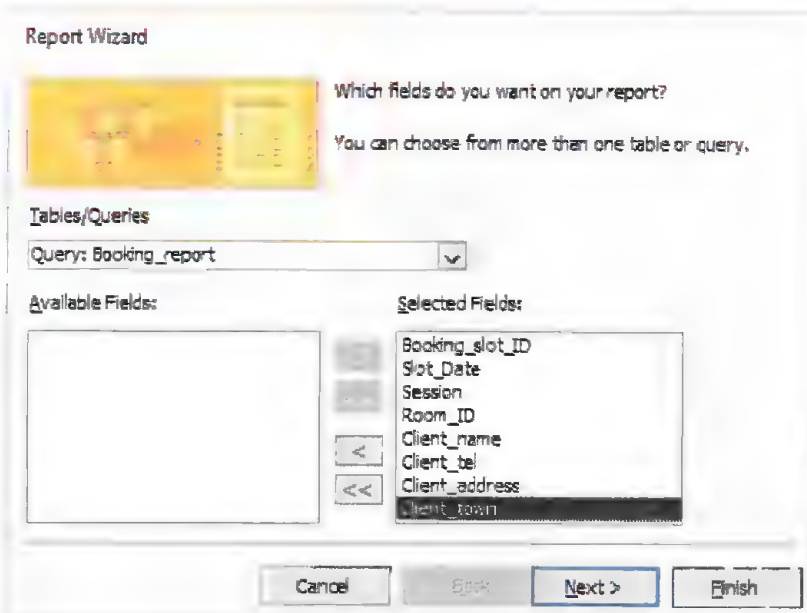


Figure 2.35: Selecting the fields for the report

Click Next and then leave the question about how you want to view your data set to 'by Booking slots'. Click Next again. You do not need to add any groups or levels so leave this option set as it is and click Next. You also do not need to sort the records in any order, so just click Next. On the next step of the wizard, choose Columnar layout, as shown in Figure 2.36.

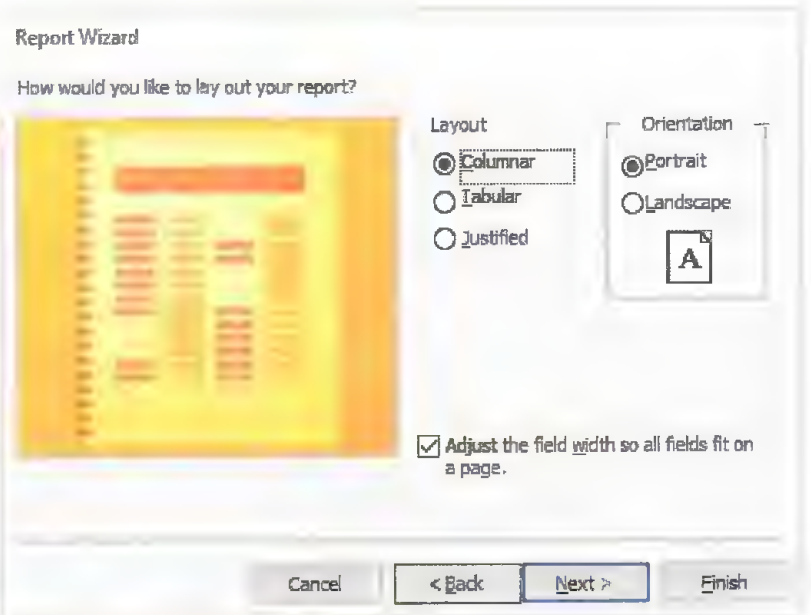


Figure 2.36: Selecting a layout for your report

Click Next again and leave the title as suggested. Then click Finish. Because the report is based on the parameter query we created earlier, you will be asked for a Booking slot ID. Enter an ID number that has been booked and you should see the completed report, as shown in Figure 2.37.

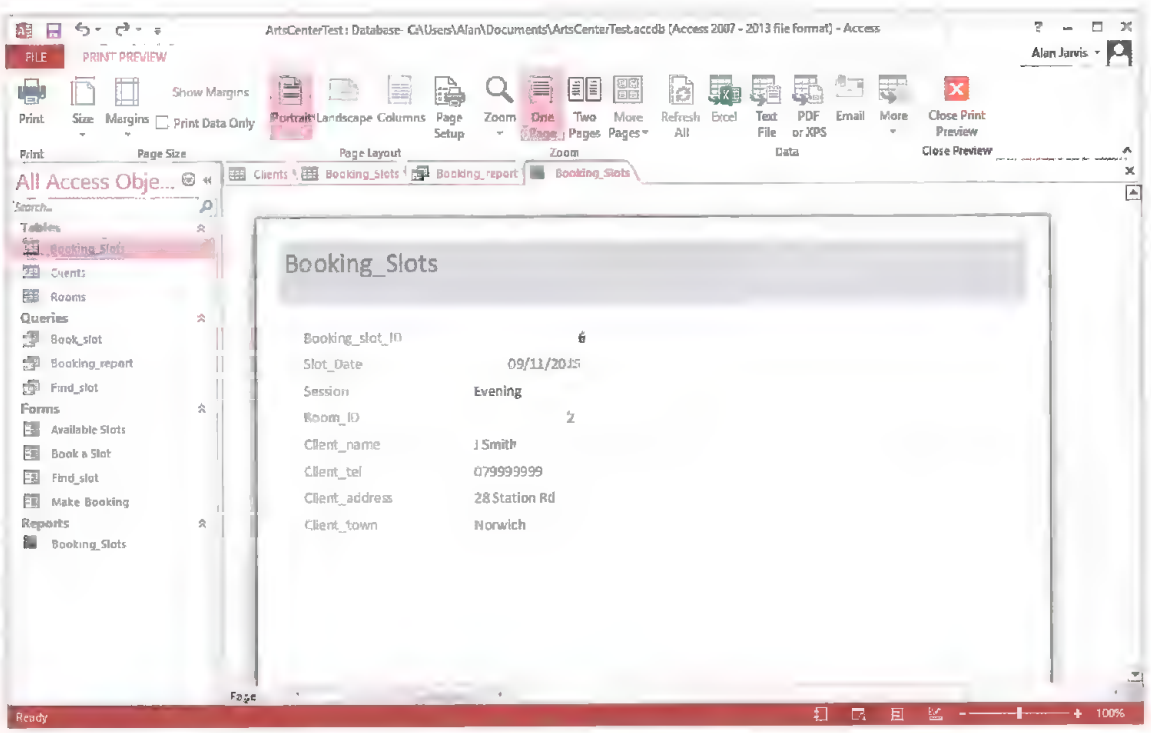


Figure 2.37: The completed report

Tip

The Report Wizard does not always set out your report in the neatest way. You can edit your report in the Report Design View and move the fields around to improve the layout.

User interface

The user interface will primarily consist of forms, and we have already looked at some examples of how to create forms and link them to tables and queries.

Two useful additional features that you can add to forms are navigation buttons and subforms.

Navigation

Navigation buttons simply provide easy-to-use buttons to navigate through the records of the table or query that a form is based on. You can add these in the Design View of a form.

Worked example: Dissford Arts Centre 14

Suppose you created a form for the Clients table of the Dissford Arts Centre database example using the wizard, as previously described. To add navigation buttons, choose the Design View for the form. Drag the form footer down a little to create some space and then, in the toolbar, click the Button icon in the Controls group and drag out a button into the space you just created. This will open the Command Button Wizard, as shown in Figure 2.38.

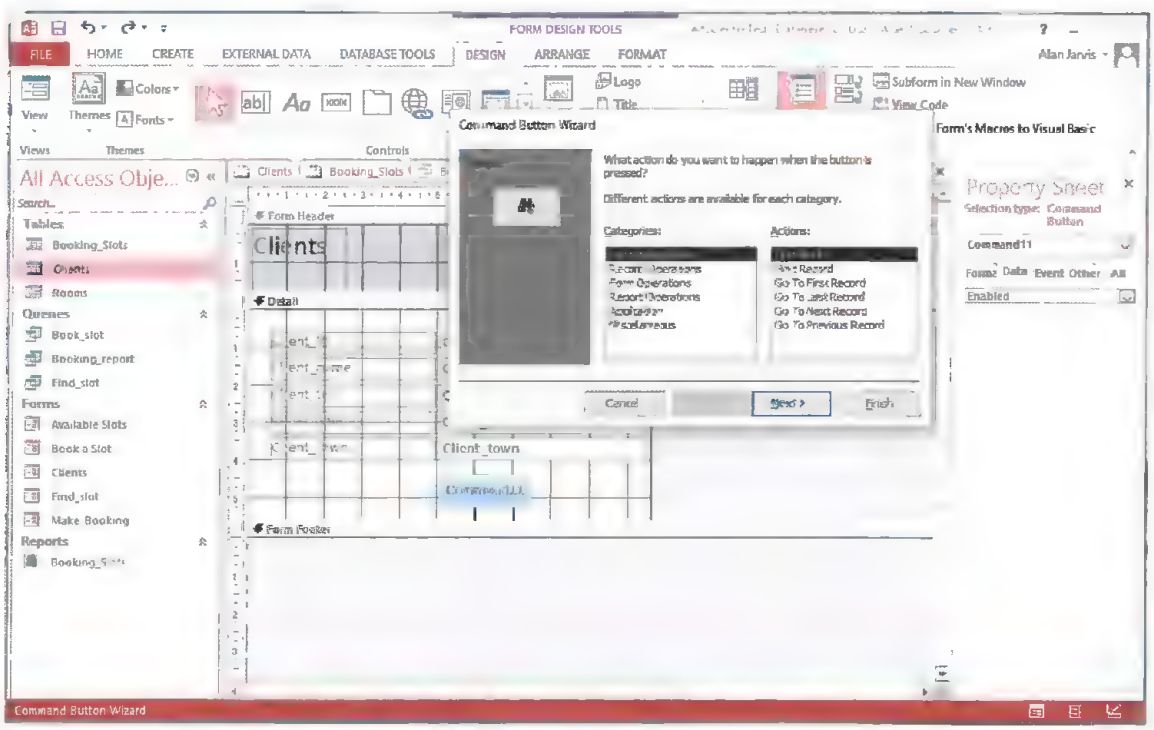


Figure 2.38: Using the Command Button Wizard

Leave the Categories list set to Record Navigation and in the Actions list choose Go to Next Record. Then click Next. Then you can choose the text or picture you want to appear on the button. Make your choice and click Finish. The Command Button will now appear on the form and, if you return to Form View, you can see and use the button you added, as shown in Figure 2.39.

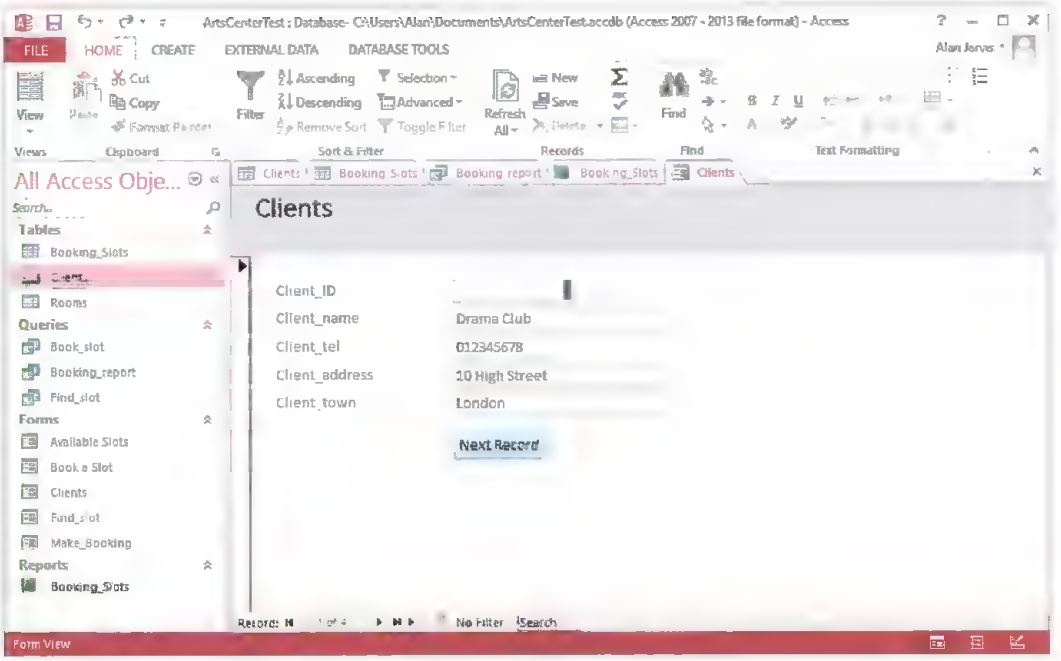


Figure 2.39: The complete navigation button

You can, of course, add more record navigation buttons, for example to go to the previous record.

Tip

There are various other buttons that you can add to forms with the Command Button Wizard, including ones that the close the form.

Subforms

Subforms are useful because they allow you to display related records from another table within the same form.

Worked example: Dissford Arts Centre 15

Using the client form to which you have just added the navigation button, you can also add a subform which will display the bookings that each client has made. Open the form in Design View again, and drag the footer bar down further to create more space under the button that you just added. Click the expand button on the right side of the Controls icon and then choose the Subform/Subreport icon (see Figure 2.40).

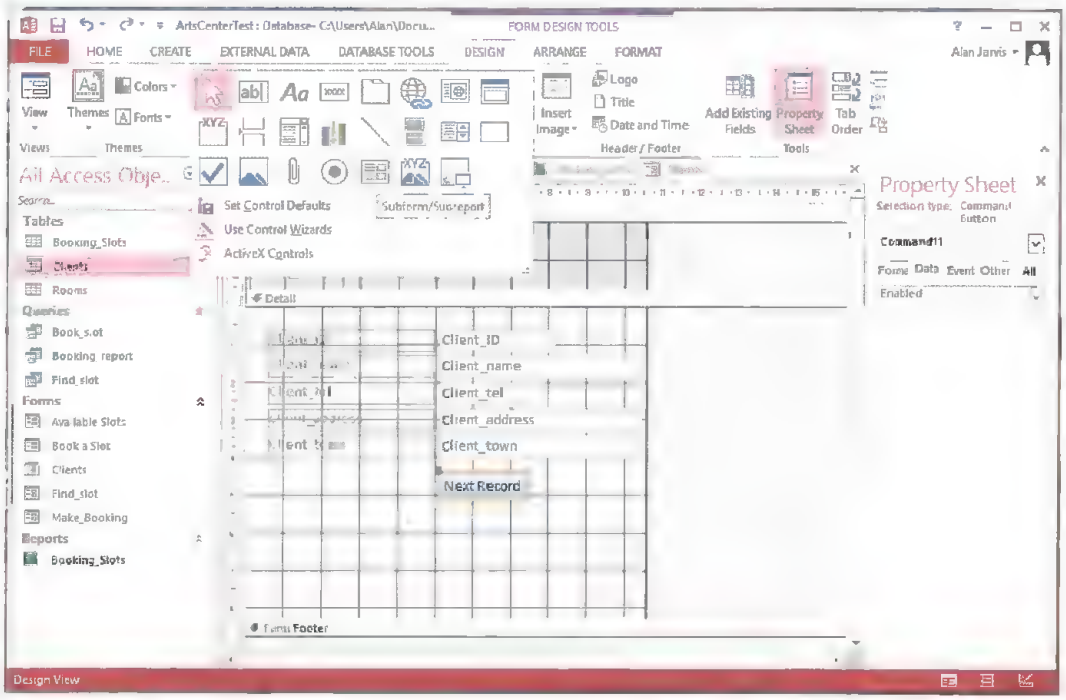


Figure 2.40: The Subform/Subreport icon

Drag out a large box under the Next Record button that you have created and the SubForm Wizard will start. In the first step, leave the 'Use existing Table or Queries' button selected and click Next. Then select the Booking_Slots table and choose all the fields (see Figure 2.41).

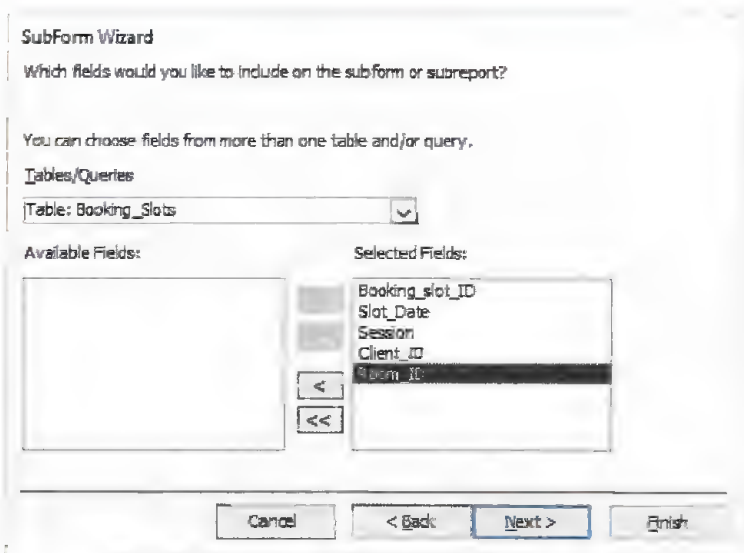


Figure 2.41: Choosing the table and fields

In the next step, Access® should have identified the link between the two tables (as they are related) (see Figure 2.42).

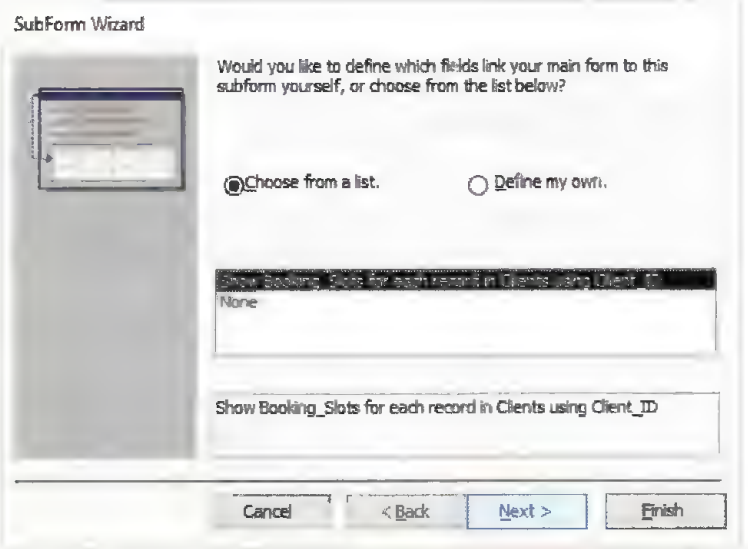


Figure 2.42: The link fields between the two tables

Click Finish. You should now see the subform box in your Design View (see Figure 2.43). You might need to drag the form footer down further to make room for the subform. Because the subform will be displayed in Datasheet View, it is best to make the box for it wider rather than longer.

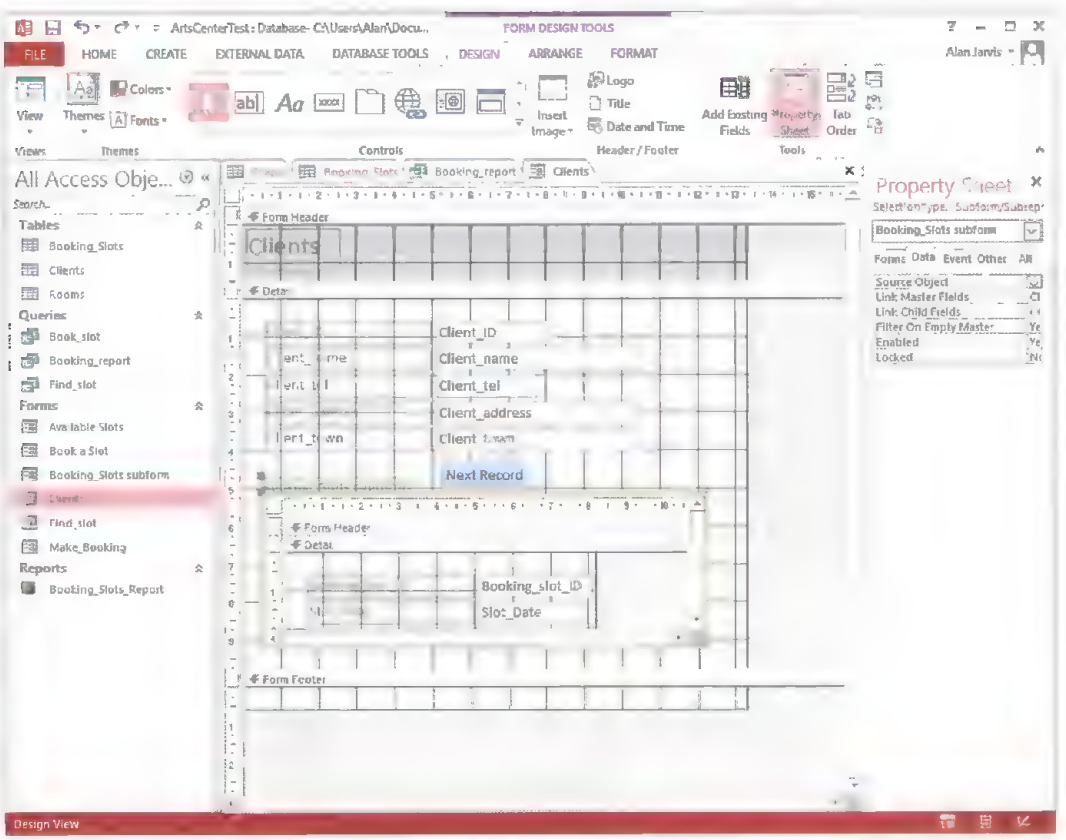


Figure 2.43: The subform box in Design View

Now change back to Form View and you should see the related booking displayed in the subform for whichever client you are viewing in the main form (see Figure 2.44).

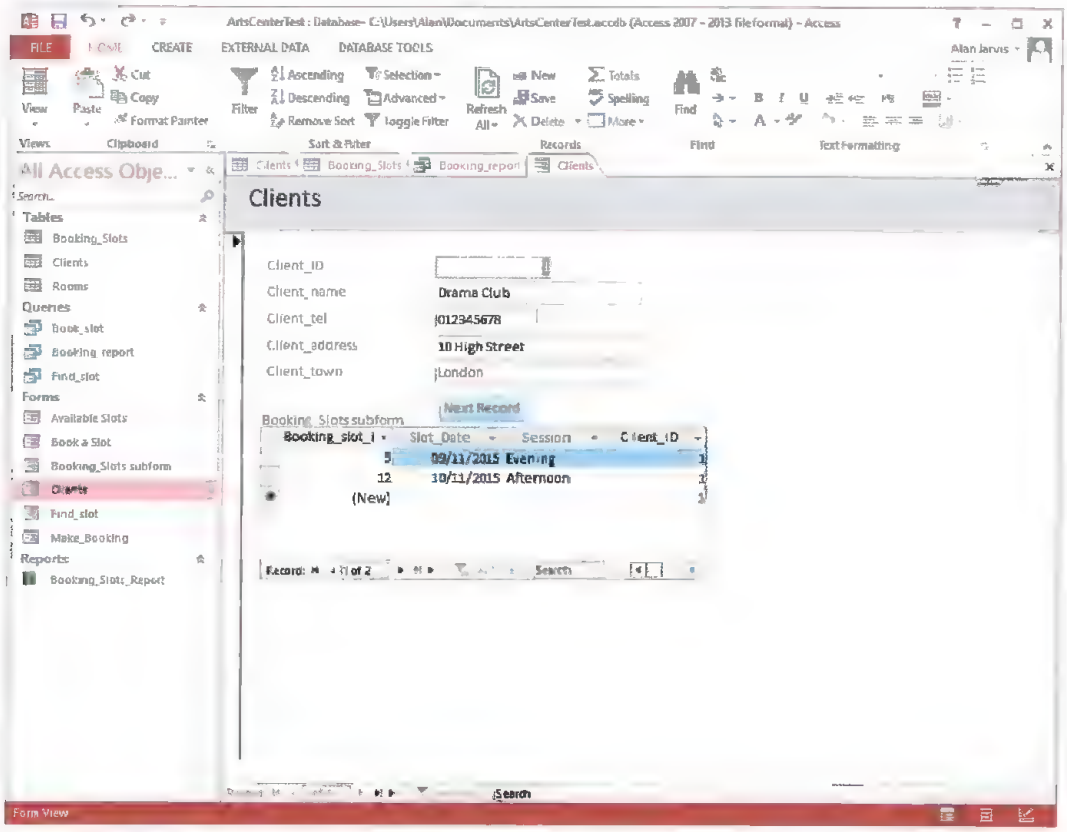


Figure 2.44: The main and subforms in Form View

Populating the database

You will need to enter or import some test data, as shown in the worked examples, in order to check the functionality of the queries, reports and forms you create. You may also need to import data, as described in the Step by step: Importing data and using an action query to manipulate it.

Link

See the Step by step: Importing data and using an action query to manipulate it.

Tip

The formatting and layout of main and subforms is quite complex and requires some practice.

Applying security levels

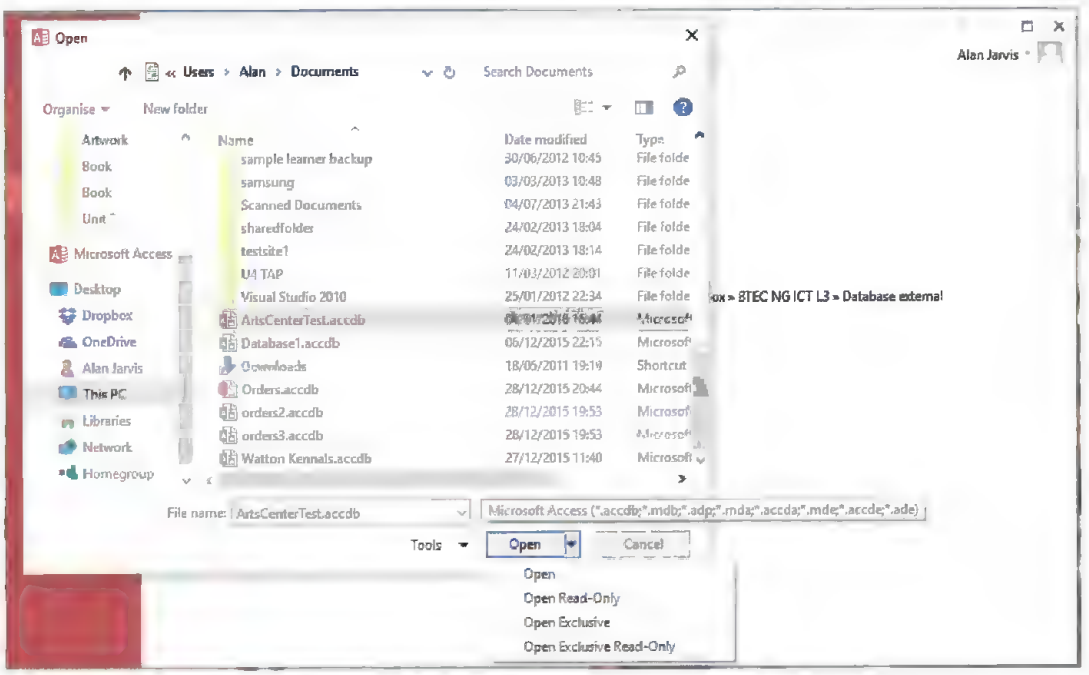
You can apply security measures to your database to control access to the data. There are two main ways to make your database secure: password protection and user access levels.

Passwords

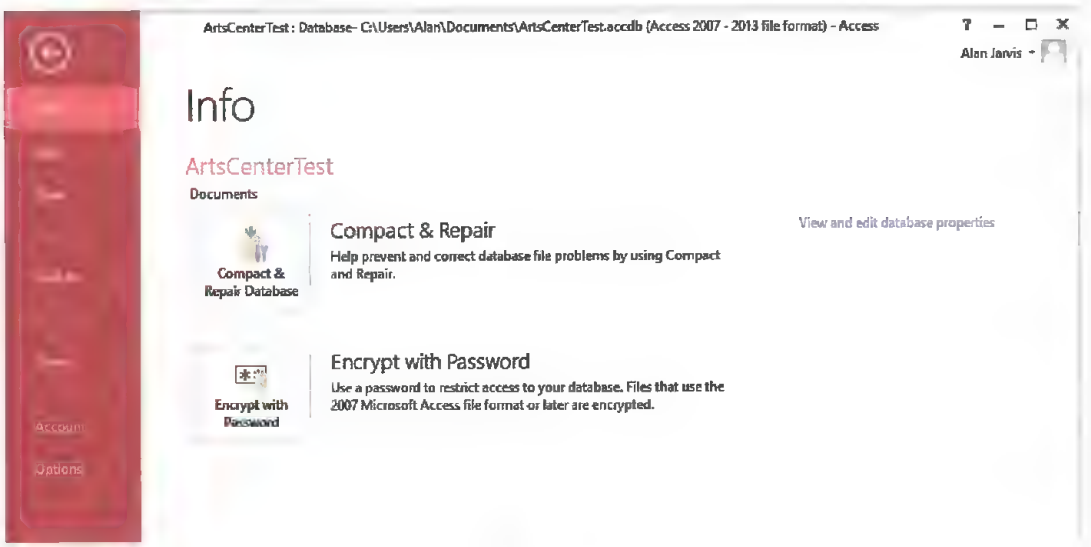
You can keep an Access® database secure by password protecting it.

Step by step: Password protecting an Access® database

1 Before you can protect a database with a password you must first open it in exclusive mode. To do this, go to the File menu, then choose Open (do not use the Recent Files option). Select the location of your database (e.g. the Documents folder) and, when the Open dialog box opens, click the down arrow on the Open button and choose Open Exclusive, as shown here.



2 Once the database has opened, choose the File menu again, and click the Encrypt with Password button, as shown here.



3 You will then be asked to enter and verify a password for the database. The database is now password protected and encrypted. DO NOT lose the password otherwise the database will be inaccessible.

User access levels

Access® database up to version 2003 allowed user-level access control. It was removed from later versions of Access (2007 and 2010, for example) as it was considered too complex.

You can still apply user-level access control using more recent versions of Microsoft® Access®, but only if you first save the database as an Access 2002–2003 version (a .mdb file).

Link

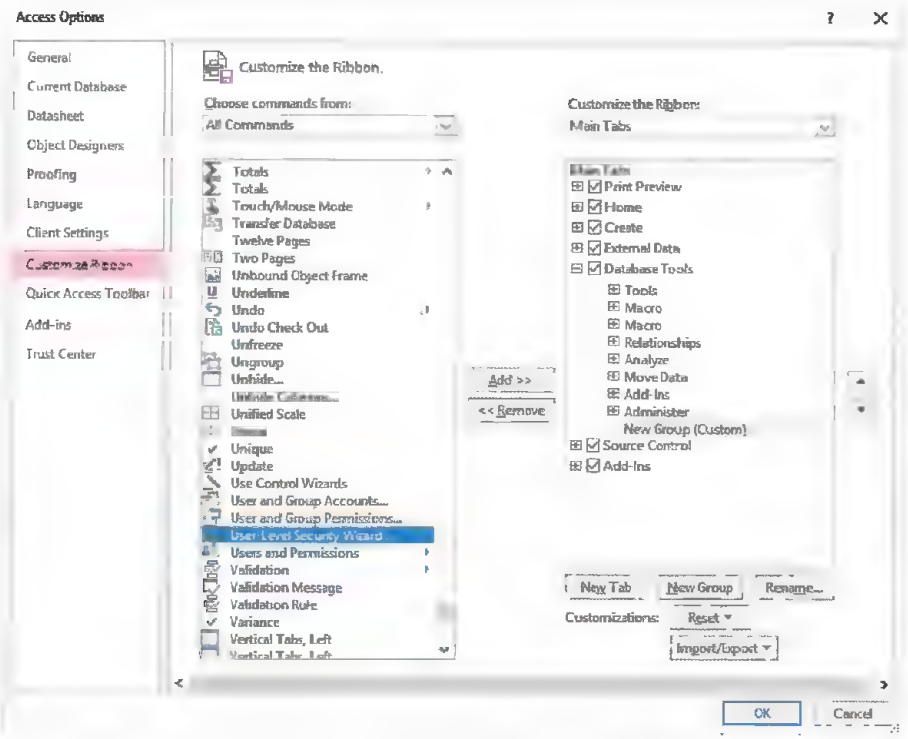
For more on database security, please go to www.databases.about.com

Step by step: Applying user access levels

4 Steps

1 To save your database as an Access 2002–2003 version (a .mdb file), so that you can apply user access level, go to the File menu, choose Save As and then select the Database file type to be Access 2002-2003 Database (.mdb).

2 Once you have done this, you will need to add the User-Level Security Wizard to the Ribbon. Go to the File menu, then choose Options (bottom of the list on the left). Once the Access Options dialog box appears, choose Customize Ribbon from the menu on the left. Click Database Tools on the right and click the New Group button, as shown here.



3 Then, from the drop-down box entitled Choose commands from (top centre left of the dialog) choose All commands. From the list underneath, scroll down and find User-Level Security Wizard (the list is alphabetical). Now click the Add button in the centre of the dialog to add the command into the new group that you previously created. Then click OK. You should now see the User-Level Security Wizard in the Ribbon under the Database Tools menu.

4 The User-Level Security Wizard takes you, step by step, through allowing you to control which users and groups of users have access to all the different database objects (such as tables, queries, forms) and what kind of access they have (such as read or modify).

Testing and refining the database solution

Once the main work of implementing the relational database solution has been completed, work can start on testing the database. It is important to test and refine a database, as you need to ensure that the end result matches the design and that it fulfils the user requirements.

Recording appropriate test documentation

The functionality of the database will be tested using the test plans created during the design phase.

Link

Example test plans are shown in the section Test plans.

Each test is carried out and the actual outcome of the test recorded next to the expected outcome in the test plan. In instances where the expected and actual outcomes differ, further investigation (including correction of any errors) is required and should be documented in the test plan.

Different types of testing

Functionality testing should cover the main features of the user interface and the underlying database structure, including referential integrity issues. Testing of the usability, accessibility and security of the database should also be completed.

This testing will require the use of appropriate test data.

Link

For more on the selection and use of appropriate test data see the section Correctness of data in the Test plans section.

Using testing outcomes to improve and refine the database solution

The database functionality testing should help to ensure that the database system is free from errors and works as expected. Any errors should be resolved and re-tested to ensure that they have been fully resolved. Usability and accessibility testing can be used to refine the database and improve its ease of use, within the constraints of time and the client's budget.

PAUSE POINT

The Access® database software has so many features that it can be difficult to remember how to use them all or to find the correct feature to do what you need. Were there things you wanted to do with the database but could not find or features that you did not know how to use?

There are lots of resources available online to learn more. See Further reading and resources.

Think of additional features that would be needed to make the Dissford Arts Centre database system fully functioninal. Try to implement some of them to develop your skills at using Access®.

Assessment practice 2.1 AO1 AO2 AO3

A local charity runs a dogs' home where they look after abandoned dogs and try to find them a new home. They have asked you to create a database of the dogs they have in the home which will allow them to search for a dog which meets the requirements of people looking to re-home one of their dogs. The facilities they need are as follows.

- Each dog's ID number, Name, Breed, Colour, Sex, Year_of_birth, Approx_ size (can be large, medium or small), Weight (in kg), Date of last vaccination and Comments (text field for a short description of the dog's history and temperament).
- The charity requires there to be a form on which to enter each dog's details.
- They require a query to allow them to search for dogs meeting a client's requirements. They should be able to input the Colour, Sex and Approx_ size and see a list of matching dogs. The results of the query should be presented in a report.
- They also need to record details of the families with which dogs have been re-homed. They need to have the details of the family (name, address, telephone number etc.) and the dog that has been re-homed with them.
- They require a form-based user interface to allow them to link a dog with the family that has re-homed the dog.

You need to:

- produce a design for these requirements including:
  - an ERD showing the relationships between entities
  - documentation describing the normalisation process for this data
  - a data dictionary for the required tables
  - user interface designs
  - designs for the queries and reports
  - test plans to test your solution.
- implement your design and using the test plans created at the design stage, test the system works as expected.
- resolve any issues and optimise your solution as a result of your testing of the database to improve the functionality and usability of the system.

Plan

- What is the task?
- What are the success criteria for this task?
- Are there any areas I think I may struggle with in this task?
- What resources am I going to need?

Do

- I have planned my time appropriately to do my research and complete the task.
- I can identify when I have gone wrong and make amendments to improve the outcome.
- I can set milestones and evaluate my progress and success at those intervals.
- I can provide a glossary of common technical terms.

Review

- I can assess what went well and what I need to do to improve.
- I can explain why I went wrong or struggled in some areas more than others.
- I can identify what I have achieved by undertaking this task.
- I know what I need to learn next.

D Evaluating a database development project

Evaluation is an important part of any project because, by evaluating how the project went, you can learn from your experiences and do better next time. You will evaluate the database design, the database created, the testing processes used and the overall success of the relational database solution.

Tip

One method that can really help you to evaluate the process of designing and creating a database is to keep notes as you go through the process. Note down things that you had problems with and how you resolved them, and also tasks that were completed with no problems. Then, when you come to the end of the process of designing, developing and testing the database, you can refer back to these notes as a reminder of how things went and the issues you had to deal with. This will make writing an evaluation much more straightforward.

Database design evaluation

The design of a database, as with any software development project, is a critical part of the process. No matter how good the creation and implementation of the database system is, if the design is flawed, then the end result is unlikely to meet the requirements.

Hopefully, the majority of design errors will be spotted during the implementation stage and the design will be corrected. Errors which only become evident once the system is implemented are often much harder to resolve. This is especially the case where large volumes of data are involved, as changes to tables or relationships will take a long time to complete if there is a lot of data already in the tables.

There are a number of things you might consider when evaluating the design of a database against the given requirements while completing or after completion of the development process.

- Use and application of an ERD – How easy was it to understand the conceptual stage of the database design? Was it clear which tables would be needed and how they were to be related? Were you able to draw up the conceptual ERD easily? Was it clear what type of relationship existed (one-to-one, one-to-many, many-to-many) between the entities?
- Normalisation – Did your conceptual design translate easily into the logical design? Were you able to verify the logical design of the table relationships with the normalisation process?
- Data dictionary – Did the data dictionary accurately and correctly list the fields, datatypes, field sizes and validation required?
- Functionality requirements and identification of any omissions – How well did the design meet the user functionality requirements? To a certain extent, this depends on how well the user requirements are defined. There might have been some areas that were omitted from the requirements, for example. How did you deal with this? You need to consider the stated purpose of the database and how the user wanted to be able to access the data. There might be stated requirements, such as that the database should be easy to use, in which case you will need to consider if your user interface designs were clear and intuitive. Were any of the requirements omitted from the database design? Some of the answers to these questions may not become evident until you have created and tested the database and its user interface.
- Design strengths and weaknesses and potential further improvements – What are the strengths and weaknesses of your design? This may also only become clear when you have implemented and tested the whole system. Weaknesses in the design are going to include those parts of the design that you had to change when it came to implementation or testing. You might also want to consider whether your design was detailed enough or if there were parts of it that did not meet the requirements or omitted any of them. What further improvements could you carry out to your database to ensure that it does meet the requirements?

Evaluation of database testing

Testing is vital to ensure the correctness and robustness of your database before it gets released it to users. You will need to evaluate the application of test data to ensure that the database solution meets requirements.

The testing of the database should reveal any errors, which should then be corrected. If the testing does not reveal any errors then it is likely that the testing has not been rigorous enough.

When evaluating database testing, there are a number of areas you should consider.

- Test data – Was your selection of test data appropriate, did you have a full range of normal, erroneous and extreme test data? Could your selection of test data have been more rigorous and extensive?
- Recording of actual results and test records – Were the results of testing fully documented? You should have created detailed test plans at the design stage. Once the database had been developed, these plans should have been used to test the database and you should have entered the actual outcomes against the expected outcomes for each test.

Tip

Where appropriate, you should take screenshots of the actual outcomes of your tests for inclusion with your analysis and evaluation of the database.

- Analysis and commenting on results – Did you make use of the actual test outcomes? Have you commented on the results and analysed them? Were there different issues that you needed to investigate? Have you done this and resolved any issues? You should also keep detailed records of the changes you have made to

resolve the issues and commented on what your tests revealed.

- Iterative processes – Once all the issues have been resolved, the whole database should be tested again, and results of the new set of tests documented. The reason for this is that the changes made to correct the issues identified in the first round of tests may have inadvertently introduced other problems. This iterative process helps to improve the quality (accuracy, readability and robustness) of the final product.
- Identifying and recording which tests were successfully met and which test data issues were not resolved – Which tests were met successfully? Were there any issues that your testing identified that you were not able to resolve? What were the reasons for not being able to resolve the issues? Constraints on time or money perhaps?

Evaluation of the database

The acid test for any software development project is whether or not it meets the requirements of the users. Every software product will have its strengths and weaknesses and, when evaluating the final database product, there are a number of things that should be considered in terms of its strengths and weaknesses.

Is it fit for purpose?

The purpose of the database should have been defined at the outset by those who commissioned the database, based on who will use the product (i.e. the client and user requirements). You can consider how well it achieves that purpose.

Given that in the assessment for this unit you will have limited time, there may well be facilities that will be needed to enable the database to achieve its purpose which you do not have time to implement. You may only be asked to complete certain facilities, so there may be things that, if the database were to be used in real life, you would need to add to make it a complete application.

Consider the Dissford Arts Centre booking system that we have been using as an example (see Worked examples throughout this unit). Although we completed the part of the system that allows staff to search for available slots and make a booking, there are a number of other facilities that would be required by a real-life database, such as the ability to modify an existing booking.

Is it easy to use?

This can be difficult for you, as the creator of the database, to judge. Remember that the users of the database will probably not be IT professionals, so things that are obvious to you may be less so for them. You might find it helpful to show your user interface designs to your peers and, if possible, to others who are not IT specialists to get their opinions on how intuitive and usable the interface is. Of course you cannot do this in the final assessment for this unit, but it might help you get some ideas about what users might find difficult to understand about interfaces, and then you can apply what you have learnt in your assessment.

Are some aspects of the system constrained by the software used?

As mentioned at the start of this unit, Microsoft® Access® is designed for use as a desktop database and it lacks many of the features required to support true multi-user environments. With a database, this is particularly limiting as data, by its nature, is best when shared with others. Access® is also not really suitable for creating databases which are accessed via the internet. Therefore, you should consider what other database software options are available to you.

How easy is it to maintain the database?

Would it be easy for someone else to update, correct and add to your database? The ease of maintenance of an Access® database will depend upon how detailed and accurate your design documentation is. This documentation should allow someone else to understand how your database has been designed and created.

Does the database meet its given requirements?

To a certain extent, this will depend on how detailed those requirements were in the first place. In many ways this question is a combination of all the questions above. Evaluation of the database against its requirements essentially requires you to ask the following question: Does the database do what it was required to do and how well does it do it?

Assessment practice 2.2

AO4 AO5

Write a justification and an evaluation of the database development project that you undertook for the dogs' home in Assessment practice 2.1.

Your justification and evaluation should consider the following questions.

- How well does the system that you developed meet the user requirements, as outlined in the scenario?
- How effective have you been in creating a high quality database solution which performs appropriately and is easy to use?
- What were the changes that you needed to make to the system during the implementation and testing phases?
- What improvements would you like to make, given more time?

Plan

- What is the task and how will my success be judged?
- What might I find difficult with this task?
- How does an evaluation and a justification differ from an explanation?
- What resources will I need?

Do

- I have collected together my notes and the work I completed for assessment practice 2.1.
- I have set task milestones and checked my progress against these.
- I have drafted out answers to the questions listed in the task.
- I have checked that what I have written provides a justification of the actions that I took in assessment practice 2.1.
- I have checked that I have evaluated the work that I completed for assessment practice 2.1.

Review

- I can assess which parts of the work went well and which parts I found difficult.
- I understand places where I might have gone wrong.
- I know what I have learnt by completing this task.
- I know what my next steps will be.

Further reading and resources

Training manuals

There are lots of free training materials on the internet including:

- Support.office.com – The official Microsoft® Office® website. It contains training material for various versions of Access®.
- docs.oracle.com/en/database/ – Oracle database documentation. It includes information on database concepts.
- **http://www.lynda.com/Access-training-tutorials/140-0.html** – Access® tutorials on Lynda.com.

Books

Connolly, T. and Begg, C. (2014) *Database Systems: A Practical Approach to Design, Implementation, and Management* (Pearson) ISBN 9781292061184
This is a comprehensive book on database theory. It is a standard university text and quite technical.

Alexander, M. and Kusleik, R. (2013) *Access 2013 Bible* (John Wiley) ISBN 9781118490358
This gives comprehensive and detailed examples of how to use all the features of Access, including advanced level ones.

Fuller, L.U. and Cook, K. (2013) *Access 2013 for Dummies* (John Wiley) ISBN 9781118516386
This is an easy-to-follow book in the Dummies' easy-to-read style.

THINK FUTURE

Hesmita Patel,

BTEC National Student

I wanted to do this course because I know that so many jobs require IT skills. I found it quite hard going, as there is a lot of technical content. I had thought that databases were pretty straightforward so it was quite a surprise that they can get quite complex. However, I did enjoy the challenge and, after a while, I began to enjoy using database software. I find it very interesting to see all the things you can do with it and I'm sure these skills will come in very handy in the business world.

I'm a bit apprehensive about the assessment, as there are lots of skills and techniques I need to be able to use for both making the design and creating the database, so I need to make sure that I am fully prepared for it.

In the future, I would like to learn more about web and server-based database systems as I think this is a really key skill for someone interested in using databases professionally.

Focusing your skills

Development plan

It is important to work out which skills you need to develop in the future. Here are some ideas to help you create a development plan.

1 Make a development plan listing the skills you currently have and those you want to develop. For the skills you need to develop, you should investigate ways of developing them and include this information in your plan.

2 Take every opportunity to develop your skills. There are lots of free resources available on the internet and lots of widely used web development tools, including database software, which is available free of charge.

3 Practise your skills by either setting up databases for your own use or look for work experience opportunities where you can practise these skills.

Getting ready for assessment

This section has been written to help you to do your best when you take the external examination. Read through it carefully and ask your tutor if there is anything you are not sure about.

About the test

The set task should be carried out under supervised conditions.

- Electronic templates for use in task activities will be provided, ahead of your assessment, for centres to download for you. These will be supplied to you at the start of your assessment.
- Work should be completed on a computer. Make sure that you have a power lead if you are using a laptop.
- Internet access is not permitted.
- During any break, materials must be kept securely by your tutor.
- You must not bring anything into the supervised environment or take anything out without your tutor's knowledge and approval.
- You should make sure that you back up your work regularly. You should save your work to your folder using the naming instructions that will be indicated in each activity.
- Remember to bring anything else you might need, such as glasses for working onscreen and refreshments.
- Turn off your mobile phone to avoid distractions!

Preparing for the test

This unit is assessed under supervised conditions. The number of marks for the unit is 66. Pearson sets and marks the task.

The external assessment will last for 10 hours in a 1 week period, and can be arranged over a number of sessions. You will be assessed on your ability to design, create, test and evaluate a relational database system to manage information.

Make sure that you arrive in good time for each test session and check that you have everything you need for the test beforehand. Make a schedule for the task to ensure that you leave yourself enough time at the end to check through your work.

Listen to, and read carefully, any instructions that you are given. Marks are often lost through not reading instructions properly and misunderstanding what you are being asked to do. Ensure that you have checked all sides of the assessment task before starting.

Proofread and correct any mistakes before handing in your work.

Key terms typically used in assessment

There are some key terms that may appear in your assessment. Understanding what these words mean will help you understand what you are being asked to do.

- The following table shows you the key terms that will be used consistently in your assessments to ensure that you are rewarded for demonstrating the necessary skills.
- Please note: the list below will not necessarily be used in every paper/session and is provided for guidance only. Only a single command word will be used per item in your test.

Key terms	Definition
Annotated screen shot	Image copy of a computer screen (obtained by pressing the print screen key then pasting into a document) with added annotations explaining what the image shows.
Database structure	The structure is composed of fields (a single piece of data, e.g. a name or date of birth), records (a complete set of fields, e.g. an employee's personnel record) and tables (a collection of records, e.g. all employees' personnel records).
Data dictionary	A centralized repository of information about data, such as meaning, relationships to other data, origin, usage, tables, fields and format.
Entity-relationship diagram (ERD)	A diagrammatical representation of database tables and their relationships to other data, origin, usage, tables, fields and format.
Evaluate	A review and synthesis of each stage of database design, (i.e. development, processes and outcomes) to provide a supported judgement about the quality. Typically, a conclusion will be required.
Normalisation	The process of organising raw data into separate related tables to minimise data redundancy.
Query	An SQL select statement which extracts data from a table or tables which match defined criteria.
Report	A database report presents information from a database. Information should be displayed simply and efficiently. Printed reports from the database should allow the viewing of information quickly and easily.
Test log	Used to plan and record program testing, record the outcomes of testing and the changes made to solve problems.
User interface	The visual part of the database through which a user interacts with a computer or software. A good interface is intuitive and allows a user to easily enter the required data accurately. A user interface is implemented using screen forms with titles, labelled boxes for data entry, buttons to perform actions and other features to make interaction as easy as possible.

A few more guidelines

- Always make a plan for your answer before you start writing. Sketch this out so that you can refer to it throughout – remember to include an introduction and a conclusion and think about the key points you want to mention in your answer. In this plan, think about setting yourself some timeframes so that you can make sure that you have time to cover everything you want to – and, importantly, have time to write the conclusion!
- Try to keep your answer as focused on your key points as possible. If you find your answer drifting away from that main point, refer back to your plan.
- Make sure that you understand everything being asked of you in the activity instructions. It might help you to underline or highlight the key terms in the instructions so that you can be sure that your answer is clear and focused on exactly what you have been asked to do.

Sample answers

Look at the sample questions which follow and the tips on how to answer them well.

Worked example

Set task brief

You have been asked to produce a database to manage cinema bookings. You have identified that the cinema has shows (with a film title, rating and description) and it has customers who want to come to see the shows.

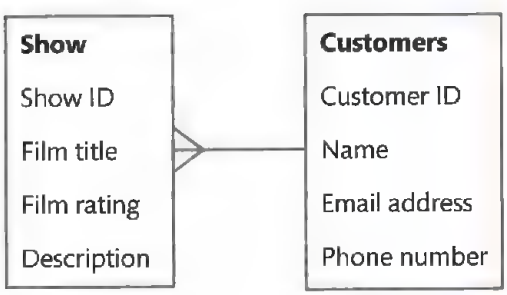
In the real assessment, you will be given a more detailed brief.

Activity 1: Entity-relationship database

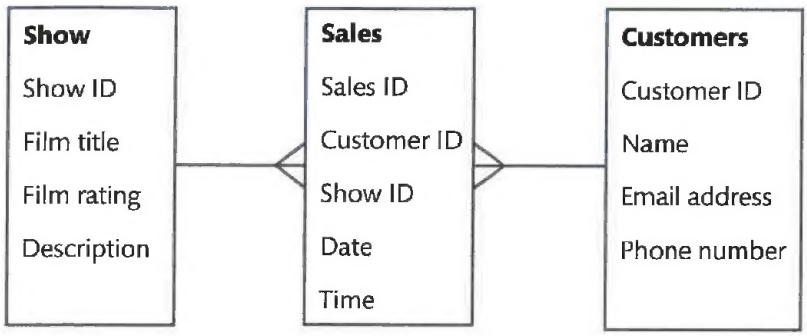
Produce an entity-relationship diagram (ERD) for the database by normalising the given data to third normal form.

Sample answer

You have drafted out an ERD which looks like this:



This shows that the Customers table is a one-to-many relationship with the Show table, as one customer can go to see many different films. However, each show has more than one customer. Lots of people go and see each showing of the film, not just one. This is, therefore, a many-to-many relationship, which needs to be resolved with a link table. Adding a Sales table to link the Show table and the Customers table will resolve the many-to-many relationship.



Activity 2: Data dictionary

Produce a data dictionary for your database using the given document.

In the real assessment you will be given a template table.

Sample answer

The first draft for the Sales table looks like this:

Data dictionary: Sales table			
Field name	**Key?**	**Datatype**	**Validation**
Sales ID	Primary	Integer	
Customer ID		Integer	
Show ID		Integer	
Date		Text, 6 characters	
Time		Text, 4 characters	

To improve this data dictionary, the Customer and Show IDs should be listed as foreign keys.

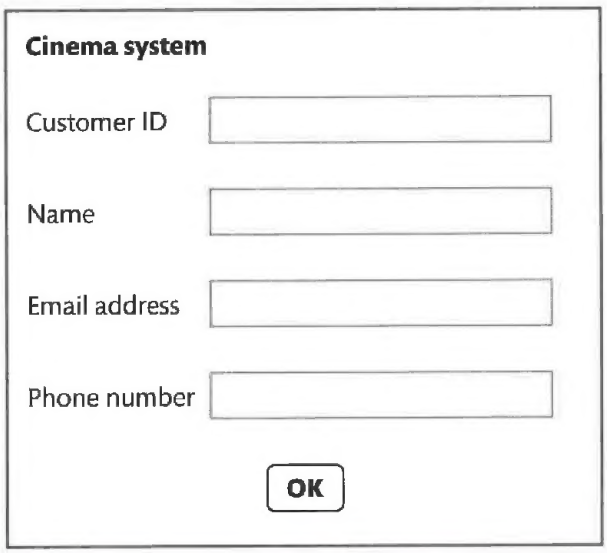
Date and time fields should have date datatypes, to ensure that they are automatically validated as the correct format for a date or time.

Activity 3: Design the user interface

For another task, you are asked to produce an interface design for the process of registering a new customer and making a sale.

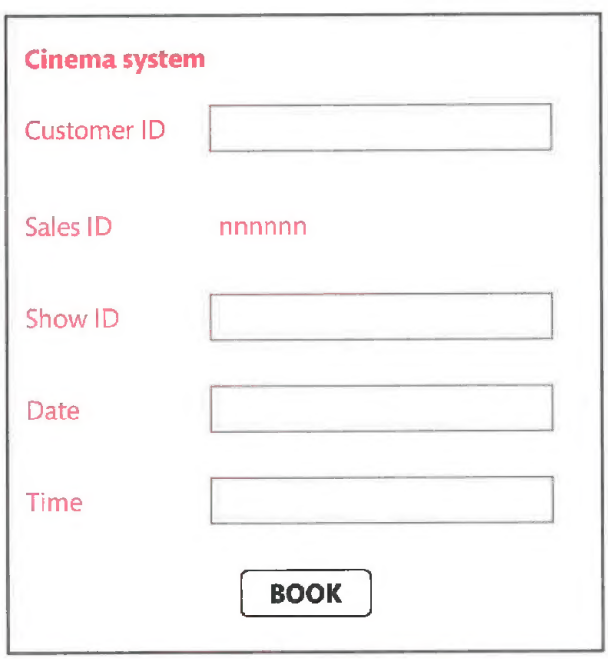
Sample answer

Your first interface design is for the customer registration:



However, the Customer ID should not be entered by the user, as it needs to be unique. It should be created automatically (an Access AutoNumber field).

The next screen is the Sales screen where the film is selected:



The Sales ID should be an AutoNumber field so that it is correctly shown as a field that the system displays rather than one that the user enters. However, the user would not know which Show ID to enter. This field would be better as a drop-down box which displays the film names.

It might also be a good idea to add a Cancel button to allow the user to back out of the transaction.

This design needs some improvement. The Customer ID would come from the previous screen so does not need to entered, just displayed.

In your assessment, you are also likely to be asked to complete other activities, such as the following.

- Activity 4: Testing plan
- Activity 5: Database development and testing (using data provided)

Activity 6: Evaluation of your database solution
Evaluate your solution.
You should consider:

- how well your solution meets the requirements of the scenario
- the quality, performance and usability of the database
- the changes made during the development and testing process.

Sample answer
This is your first draft:

I have worked very hard on this database and I am really happy with the result. Most of it works well and it does what it is supposed to do and could be used to book cinema tickets with a few additions. I think the user interface is ok but perhaps it needs a few more buttons to make it easier to use, like some 'Back' or 'Cancel' buttons on some forms. The report I produced could also be made better by tidying it up a bit. I didn't have a lot of problems creating the database. There were a few things I found difficult but I managed to sort them out.

The main problem with this evaluation is that is it too vague. For example, it says you are happy with the database but does not say why. You would need to say something like:

I'm happy with the database design as the tables are all related correctly and there is no duplication of data. I checked the design with the normalisation process. The data dictionary is also fully complete with appropriate data types and validation for all the fields where it can be used. This should help make sure that correct data is input, wherever possible. The queries I developed extract the correct data and have all the required fields and provide the forms and reports with the data that they require.

The forms are laid out reasonably well but would benefit from some of the labels and fields being adjusted in size and position, and including Cancel buttons would make them easier to use.

You would also need to provide more detail about how the development process went and the problems you had and how you overcame them. This is one reason why keeping a diary during the development process, in which you can note down these things, can really help when it comes to writing the evaluation.

Remember that when writing an evaluation you need to give specific details about:

- what you think is good (and why it is good) or went well with your database
- what you had difficulty with and how you overcame those difficulties
- what you would improve or correct given more time.